

SOMMAIRE

RESUME EXECUTIF	1
INTRODUCTION (Contexte, hypothèses, exigences)	2
A-Architecture cible	3
B-Détails de conception	4
C-Sécurité (IAM, chiffrement, réseau, conformité)	6
D-Déploiement (IaC, CI/CD, environnements, rollback)	6
E- Observabilité & opérations (dashboards, alertes, runbooks)	17
F-Tests & résultats (méthode, outils, mesures, limites)	18
G-Coûts & optimisation (FinOps)	18
CONCLUSION (Risques, limites, perspectives)	19
REFERENCES & ANNEXES	20

RESUME EXECUTIF

L'architecture technique est une vue tournée sur l'organisation logique de la plateforme informatique, c'est-à-dire les moyens techniques clés qui seront utilisés par tous les logiciels applicatifs. La vue contient le matériel informatique, les logiciels systèmes, les middlewares, ainsi que les réseaux de télécommunication et les relations entre ces différents éléments. Pour une entreprise ou une institution, le choix de l'architecture technique vise à maximiser les possibilités d'implantation de logiciels du commerce ainsi que de réalisation de logiciels sur mesure. Il vise également à rentabiliser l'utilisation du matériel et des logiciels déjà acquis par l'institution. Pour une entreprise ou une institution qui transforme son architecture technique, le plan d'architecture est accompagné d'un planning et d'un budget des acquisitions, des ventes et des opérations de migration nécessaires pour aligner le système informatique avec le plan. L'architecture logicielle est une vue tournée sur l'organisation interne et le découpage d'un logiciel en modules.

INTRODUCTION (Contexte, hypothèses, exigences)

Le marché tchadien des jeux vidéo est de plus en plus croissant de marques telles que Sony, Microsoft importées pour l'essentiel de Dubaï (Emirats Arabes Unis) qui constituent l'écosystème des jeux vidéo importés. La société de jeux souhaite faire la promotion de ses services en ligne via une application sécurisée.

A cet effet, elle sollicite notre expertise.

Compte tenu des enjeux, nous présentons quelques hypothèses suivantes :

- Une solution sécurisée conçue avec le Framework Angular et avec un langage informatique tel que JavaScript.
- Une solution sécurisée conçue avec le Framework Symfony et avec un langage informatique tel que PHP.

Le contenu de notre mémoire comporte plusieurs parties à savoir l'Architecture cible, ensuite les Détails de conception, ensuite la Sécurité (IAM, chiffrement, réseau, conformité), ensuite le Déploiement (IaC, CI/CD, environnements, rollback), ensuite les Observabilité & opérations (dashboards, alertes, runbooks), ensuite les Tests & résultats (méthode, outils, mesures, limites), ensuite les Coûts & optimisation (FinOps), ensuite les Risques, les limites, les perspectives et enfin les Références & annexes.

A- Architecture cible :

A travers le développement d'une application Front end avec le Framework Angular.

Le cahier de charges

- **Présentation de la société**

Présentation de l'activité de la société : L'activité principale de la société «GUEGUE IT» est de vendre des consoles de Playstation 4, de XBOX ONE et des CD de jeux vidéo. Expression des besoins : Besoin de réalisation d'une petite application afin de l'utiliser pendant la période de promotion fixé pour les vacances d'été entre mai et août. Explication du rôle de l'application dans la stratégie de la société : Le rôle de l'application dans la stratégie de la société est de passer l'information sur les prix des CD de jeux vidéo pendant la période de promotion. Conception : Conception de la solution et identification des risques. Indication de la date butoir pour la fin du projet : Mise en ligne de l'application escomptée après le 18/05/2025.

- **Cible à laquelle s'adresse l'application**

Cible : La cible toute personne souhaitant acheter des CD de jeux vidéo moins chères. Lieu de la cible : De manière générale, la cible se trouve sur internet, à travers des canaux numériques qu'elle fréquente tels que les réseaux sociaux.

- **Aspects d'ergonomie et de graphisme**

Couleurs principales de l'application : Les couleurs principales de l'application sont le vert et le blanc. Utilisation du logo : Le logo utilisé est celui de la société, au niveau du côté gauche de l'entête. Police de caractère mise en place sur l'application : La police de caractère mise en place sur l'application est diverse et variée. Design ou effets particuliers : Pour le cas du design ou des effets particuliers, nous utiliserons HTML 5 pour le contenu, CSS 3 pour la présentation, Bootstrap 5 pour rendre l'application responsive et TypeScript (JavaScript) pour amener des animations sur l'application.

- **Aspect fonctionnel de l'application**

L'application requiert une page web où le visiteur peut visualiser les différents CD de jeux vidéo de PS4 et XBOX ONE qui sont à vendre. Au niveau de la présentation de chaque CD, nous avons une petite fonction d'incrément et de décrémentation au niveau du nombre afin de voir le coût, sachant également que tous

ces CD ont le même prix. Et au niveau du pied de page, nous avons le contact du vendeur.

- **Aspects techniques de l'application**

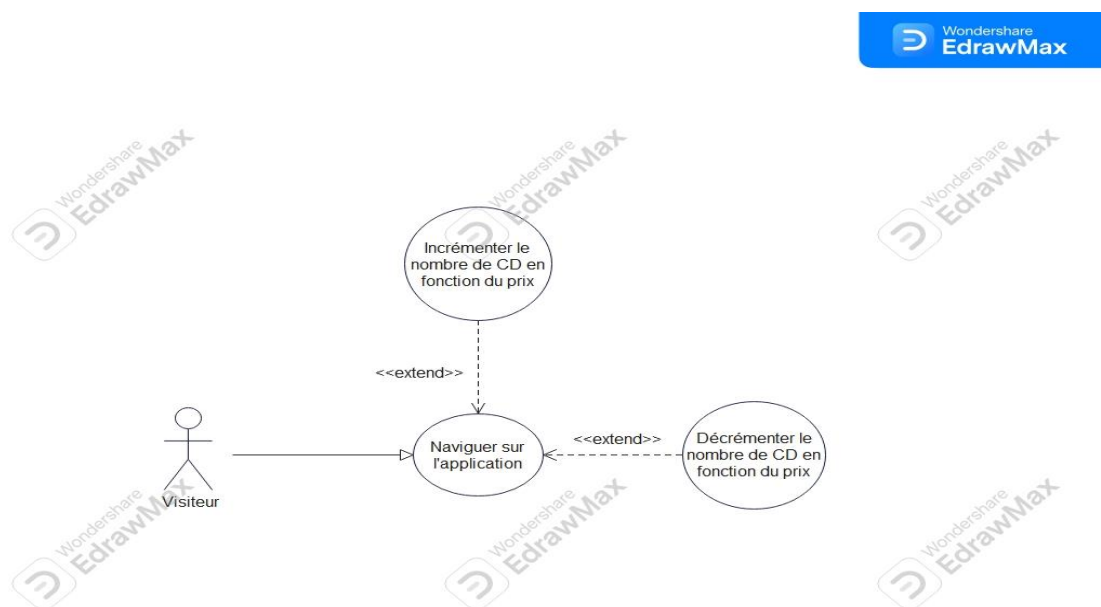
Solution technique : La solution technique que nous privilégions pour le développement de l'application est l'utilisation du Framework Angular de Google à travers le logiciel open source Visual Studio Code qui est un environnement de développement intégré ou IDE. Cas du nom de domaine et de l'hébergement : Le nom de domaine et l'hébergement s'obtiendra avec Firebase de Google qui est un ensemble d'outils pour l'hébergement et le développement d'applications mobiles et web.

B- Détails de conception :

Modélisation UML

UML (**U**nified **M**odeling **L**anguage ou Langage de modélisation unifié) est un langage de modélisation fondé sur le concept objet. L'objectif d'UML est de fournir une notation standard utilisable dans le développement des systèmes informatiques basés sur l'objet.

- **Diagramme de cas d'utilisation Visiteur**



Fiche de Cockburn user case

Fiche de Cockburn user case


- Titre : Diagramme de cas d'utilisation Visiteur.
- Résumé : Nous avons la représentation du visiteur qui peut naviguer sur l'application, en faisant l'incrémentation et la décrémentation du nombre de CD en fonction du prix.
- Acteur : Visiteur.
- Motivation : L'acteur veut naviguer sur l'application.
- Pré-condition(s) : Aucune.
- Post-condition(s) : Aucune.
- Exceptions : Aucune.
- Remarques ergonomiques (éventuellement) : Aucune.
- Contraintes non fonctionnelles (éventuellement) : Aucune.
- Scénario Nominal : [Enchaînement de Cockburn].

🚦 Sécurité dans Angular

Angular intègre de manière native des processus avancés de protection contre les attaques les plus répandues connues sous le nom de Cross-Site Scripting (XSS), ainsi que des attaques sous les noms d'injection SQL et de Cross Site Request Forgery (CSRF).










🚦 Présentation de l'application

Notre application est disponible en ligne à travers le lien suivant : <https://angularguegueit-app.web.app/>



GUEGUE IT

Les CD de jeux vidéo en promotion chez GUEGUE IT

 <p>Call of Duty Modern Warfare III Prix : 1 CD vendu entre 50 Euros et 60 Euros Nombre : 1</p>	 <p>Gran Turismo 7 Standard Edition PS4 Prix : 1 CD vendu entre 50 Euros et 60 Euros Nombre : 1</p>	 <p>Marvel's Avengers PS4 Prix : 1 CD vendu entre 50 Euros et 60 Euros Nombre : 1</p>
 <p>Dark Souls Trilogy - PS4 Prix : 1 CD vendu entre 50 Euros et 60 Euros Nombre : 1</p>	 <p>Bandai Namco Entertainment Elden Ring - Standard Edition PS4 Prix : 1 CD vendu entre 50 Euros et 60 Euros Nombre : 1</p>	 <p>EA SPORTS FC 25 Standard Edition XBOX Series X S / XBOX One Jeu Vidéo Français Prix : 1 CD vendu entre 50 Euros et 60 Euros Nombre : 1</p>
 <p>EA SPORTS FC 25 Standard Edition PS4 Jeu Vidéo Français Prix : 1 CD vendu entre 50 Euros et 60 Euros Nombre : 1</p>	 <p>Ghost Recon: Breakpoint - Edition Gold PS4 Prix : 1 CD vendu entre 50 Euros et 60 Euros Nombre : 1</p>	 <p>Cobra Kai 2 Dojos rising PS4 Prix : 1 CD vendu entre 50 Euros et 60 Euros Nombre : 1</p>

Contact : +237690539407

Copyright © 2025 GUEGUE IT - Tous droits réservés

C- Sécurité (IAM, chiffrement, réseau, conformité) :

Politique associée : dev.json

Dev User a les permissions suivantes : Upload / Download de fichiers ; Consultation que de son bucket, créer un bucket supplémentaire ; Il a le droit de supprimer les buckets qu'il a créé ; Ils ne voient pas ceux des autres (ceux qu'il n'a pas créé) et Il utilise S3 / EC2 (droit ec2).

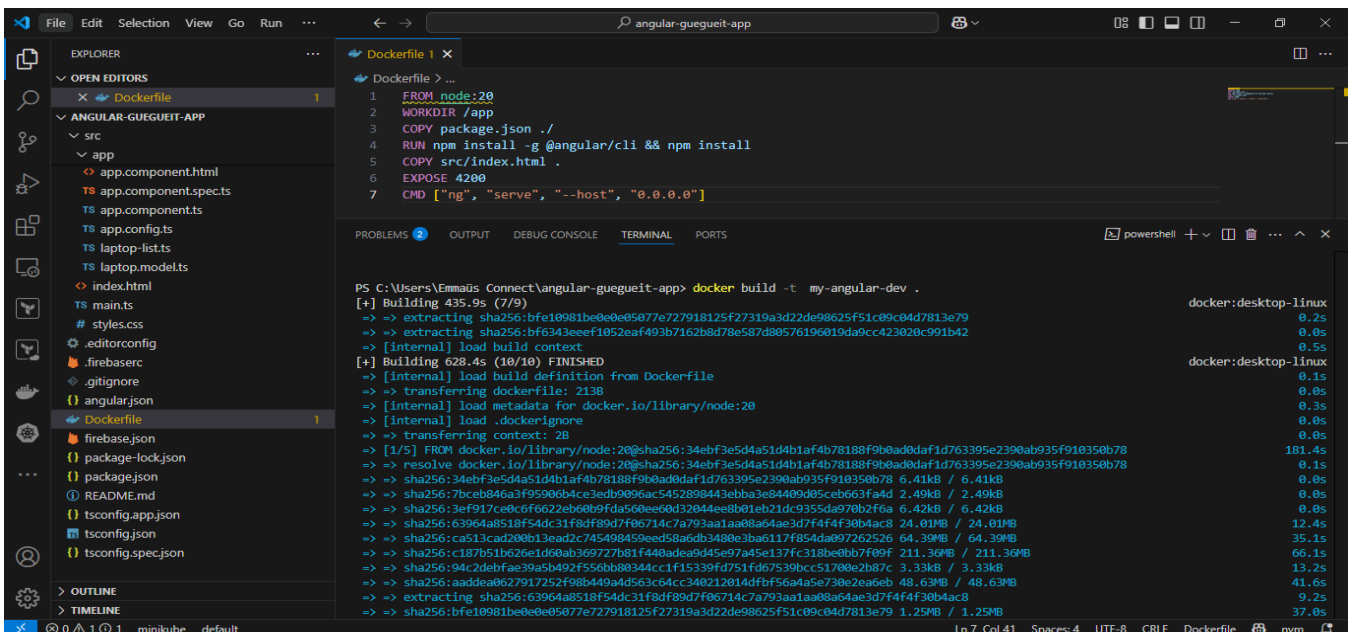
Nous avons le code json disponible sur le référentiel du code en archive .zip

D- Déploiement (IaC, CI/CD, environnements, rollback) :

Création de nœuds avec Minikube

• Etape 1 :

Nous allons tout d'abord créer le Dockerfile et taper la commande "docker build -t my-angular-dev ." qui nous permet de créer une image Docker à partir du Dockerfile situé dans le répertoire courant (.).



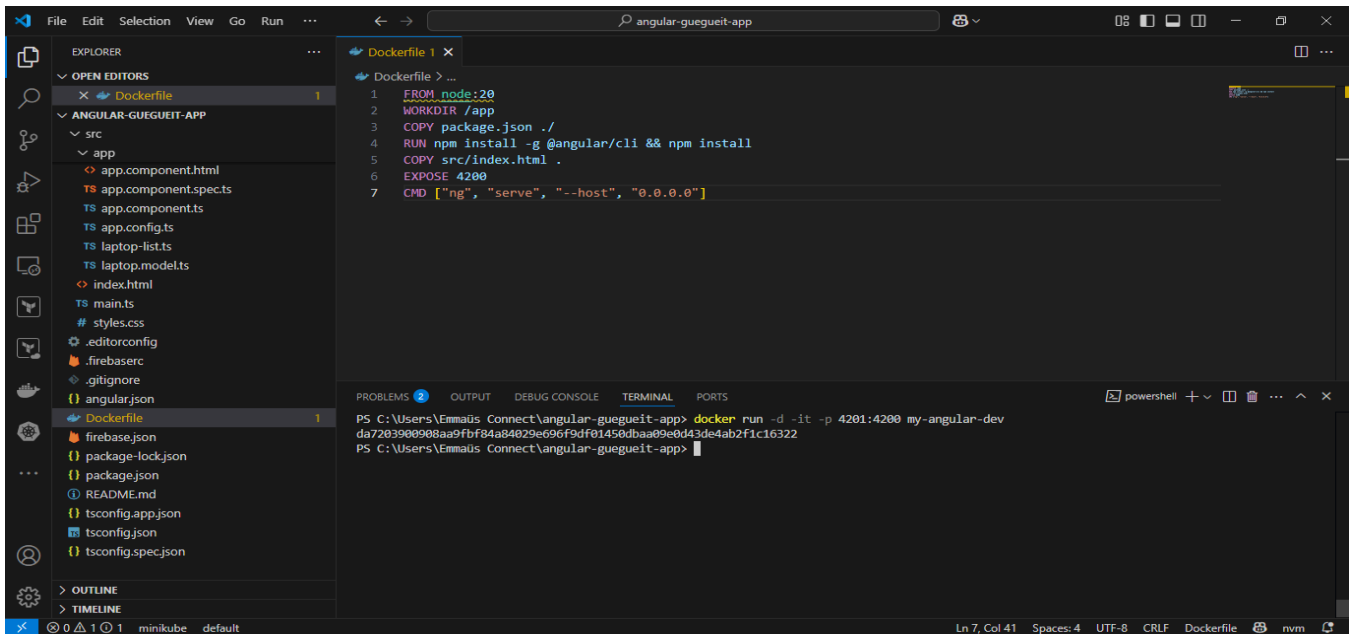
```
File Edit Selection View Go Run ... angular-guegueit-app
EXPLORER
  OPEN EDITORS
    Dockerfile
  ANGULAR-GUEGUEIT-APP
    src
    app
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.config.ts
      laptop-list.ts
      laptop.model.ts
      index.html
      main.ts
      styles.css
      .editorconfig
      .firebaserc
      .gitignore
      angular.json
      Dockerfile
      firebase.json
      package-lock.json
      package.json
      README.md
      tsconfig.app.json
      tsconfig.json
      tsconfig.spec.json
    OUTLINE
    TIMELINE
  minikube default

Dockerfile
1 FROM node:20
2 WORKDIR /app
3 COPY package.json ./
4 RUN npm install -g @angular/cli && npm install
5 COPY src/index.html .
6 EXPOSE 4200
7 CMD ["ng", "serve", "--host", "0.0.0.0"]

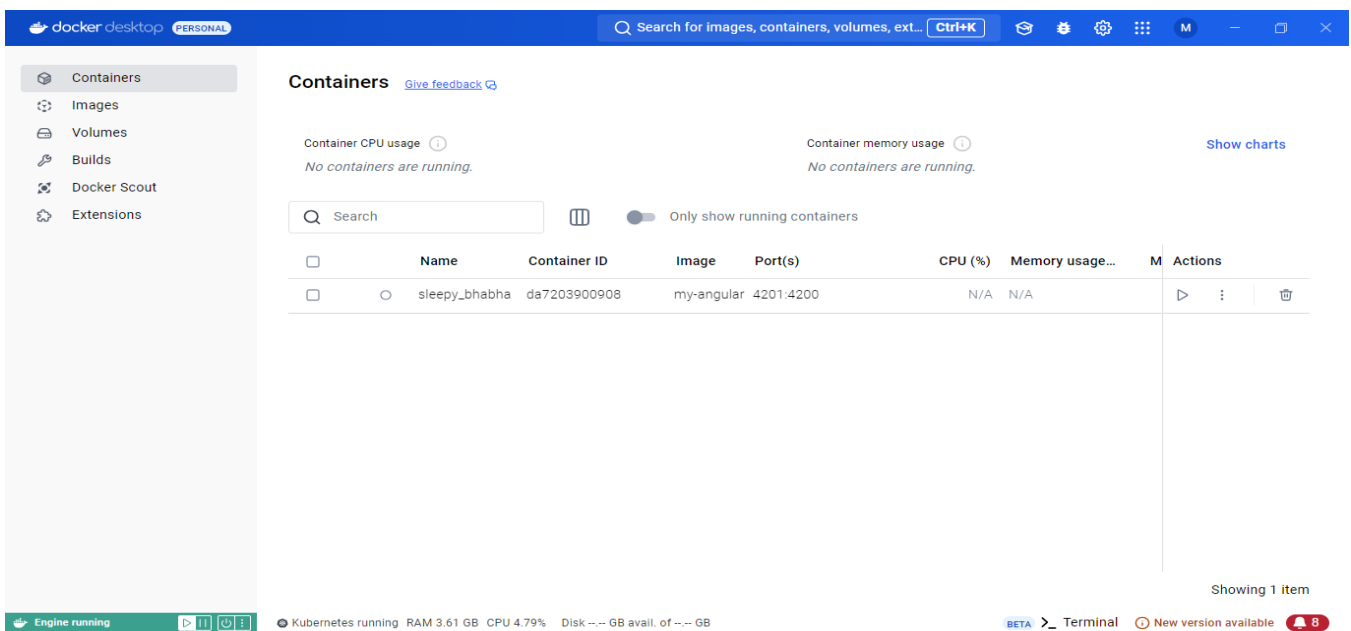
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Emmaüs Connect\angular-guegueit-app> docker build -t my-angular-dev .
[+] Building 435.9s (7/9)
-> extracting sha256:bfe10981be0e0e5077e727918125f27319a3d22de98625f51c09c04d7813e79 0.2s
-> extracting sha256:bfe10981be0e0e5077e727918125f27319a3d22de98625f51c09c04d7813e79 0.0s
-> [internal] load build context 0.5s
-> [internal] load build definition from Dockerfile 0.1s
-> [internal] load metadata for docker.io/library/node:20 0.3s
-> transferring dockerfile: 213B 0.0s
-> [internal] load .dockerignore 0.0s
-> transferring context: 2B 0.0s
-> [1/5] FROM docker.io/library/node:20@sha256:34ebf3e5d4a51d4b1af4b78188f9b0ad8daf1d763395e2390ab935f910350b78 181.4s
-> resolve docker.io/library/node:20@sha256:34ebf3e5d4a51d4b1af4b78188f9b0ad8daf1d763395e2390ab935f910350b78 0.1s
-> sha256:34ebf3e5d4a51d4b1af4b78188f9b0ad8daf1d763395e2390ab935f910350b78 6.41kB / 6.41kB 0.0s
-> sha256:7bceb846a3f95906b4ce3ed90996ac5452898443ebba3e84409d05ceb663fa4d 2.49kB / 2.49kB 0.0s
-> sha256:3ef917ce0c6f6622eb0b9fda560ee60d32044ee8b01eb21dc9355da970b2f6a 6.42kB / 6.42kB 0.0s
-> sha256:63964a8518f54dc31f8df89d7f06714c7a793aa1aa08a64ae3d7f4f4f30b4ac8 24.01MB / 24.01MB 12.4s
-> sha256:ca513cad200b13ead2c745498459eed58a6db3480e3ba6117f854da097262526 64.39MB / 64.39MB 35.1s
-> sha256:c187b51b62e1d60ab369727b81f440adea9d45e97a45e137fc318be0bb7f09f 211.36MB / 211.36MB 66.1s
-> sha256:94c2debfae39a5b492f556bb80344cc1f15339fd751fd67539bcc51700e2b87c 3.33kB / 3.33kB 13.2s
-> sha256:aaddea0627917252f98b449a4d563c64cc340212014dfbf56a4a5e730e2a6eb 48.63MB / 48.63MB 41.6s
-> extracting sha256:63964a8518f54dc31f8df89d7f06714c7a793aa1aa08a64ae3d7f4f4f30b4ac8 9.2s
-> sha256:bfe10981be0e0e5077e727918125f27319a3d22de98625f51c09c04d7813e79 1.25MB / 1.25MB 37.0s
```

• Etape 2 :

Ensuite, nous allons taper la commande "docker run -d -it -p 4201:4200 my-angulardev" qui nous permet d'exécuter un conteneur à partir de l'image Docker nommée «myangular-dev». Nous avons donc notre conteneur appelé «sleepy_bhabha» présent sur notre Docker Desktop.

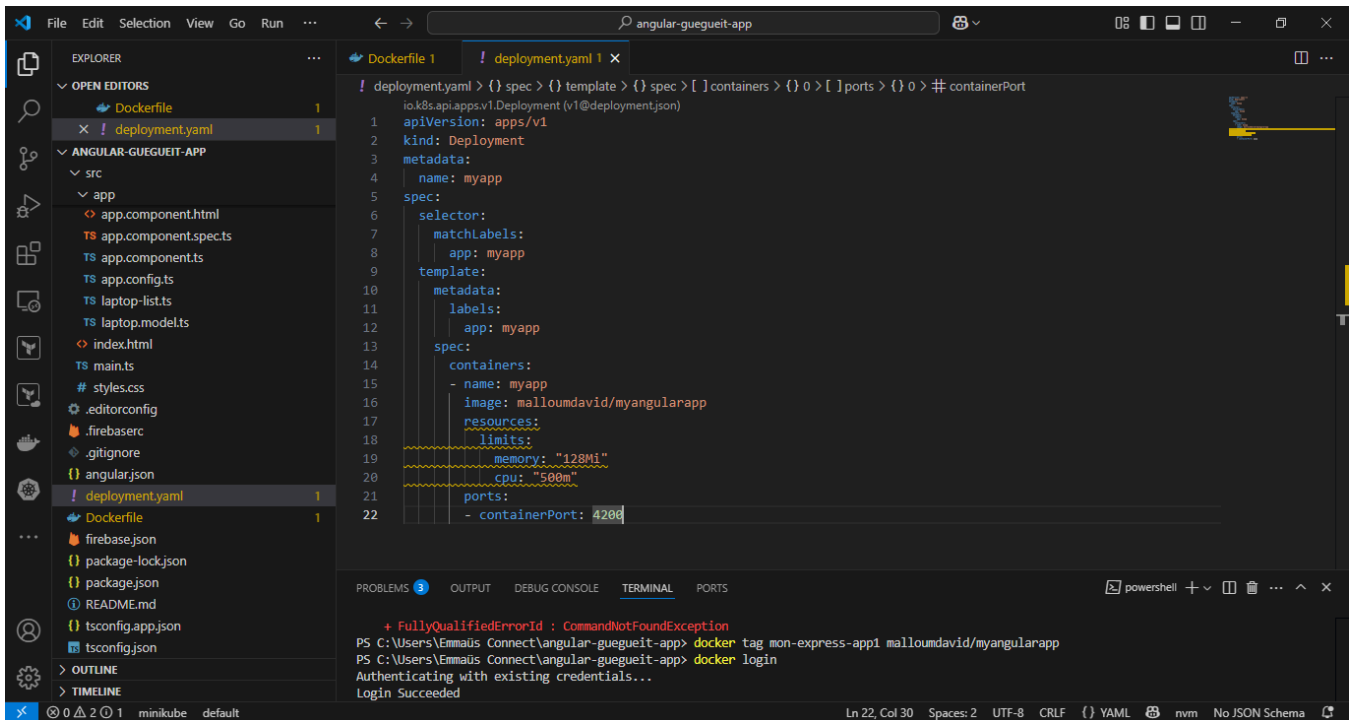


Nous avons donc notre conteneur appelé «sleepy_bhabha» présent sur notre Docker Desktop.



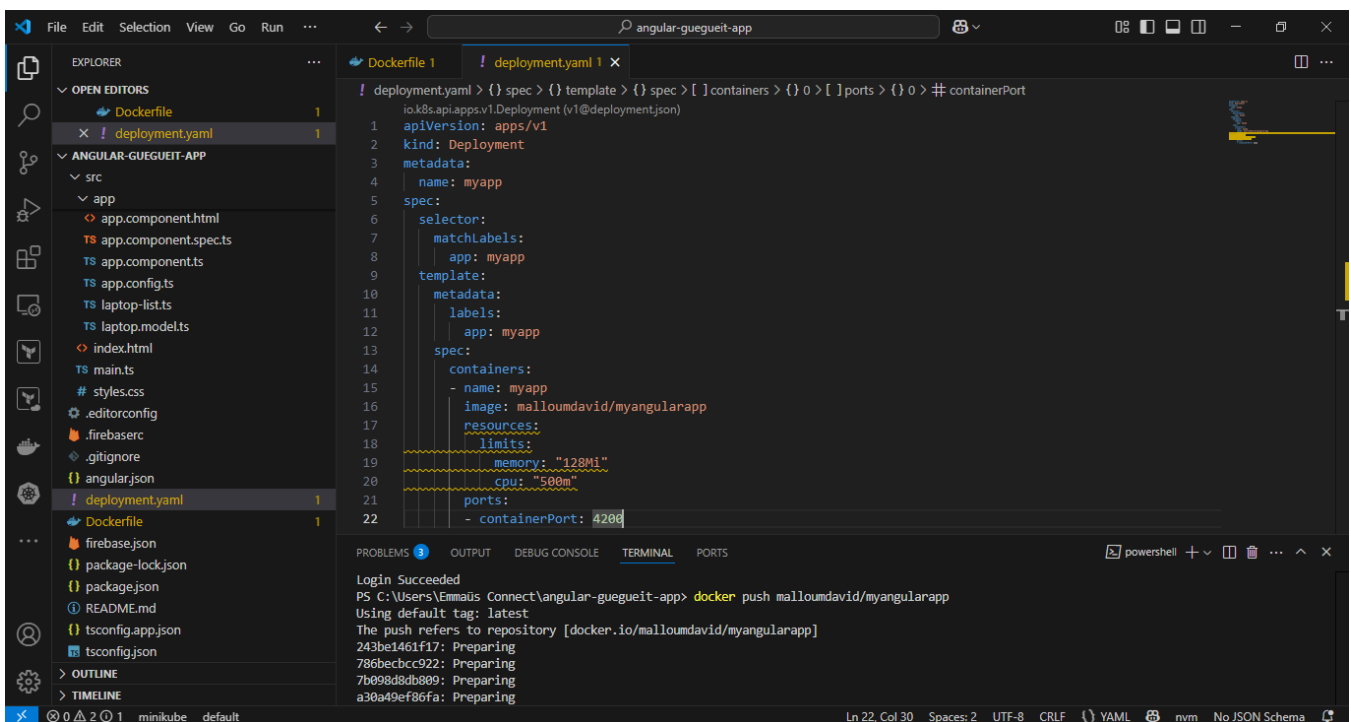
• Etape 3 :

Ensuite, nous allons créer le fichier deployment.yaml et taper les commandes "docker tag mon-express-app1 malloumdavid/myangularapp" qui nous permet d'ajouter un nouveau nom (tag) à une image Docker existante et "docker login" qui nous permet de nous authentifier auprès de notre Docker Hub.

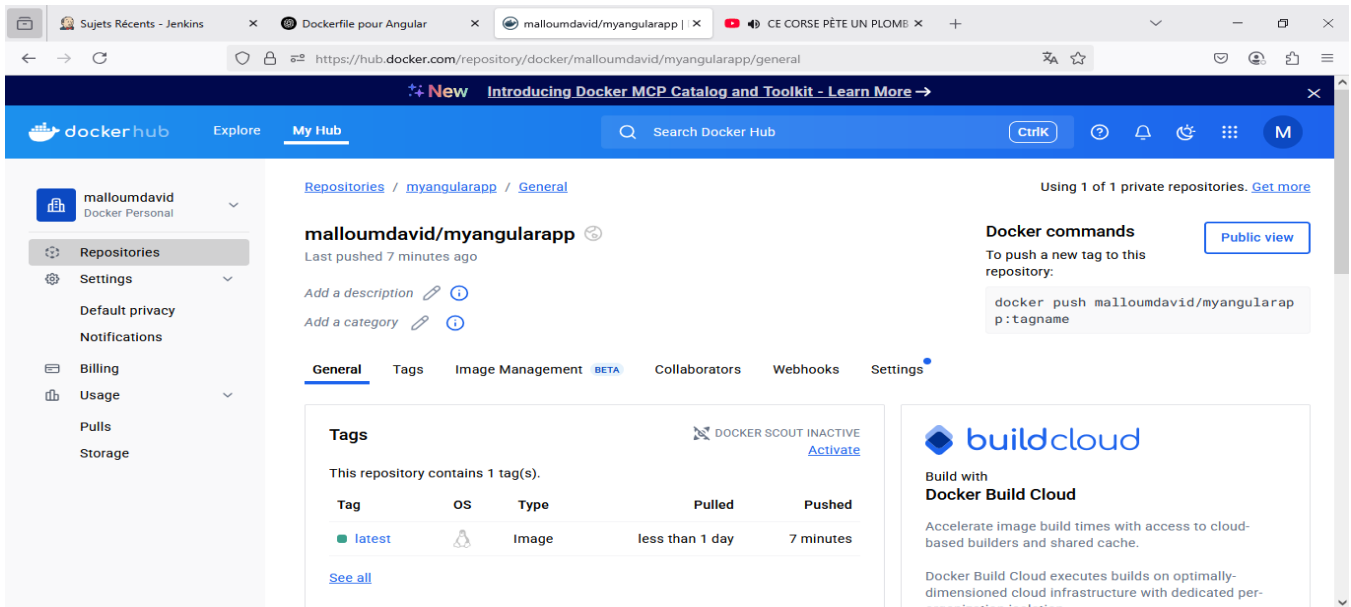


• Etape 4 :

Ensuite, nous allons taper la commande "docker push malloumdavid/myangularapp" qui nous permet d'effectuer le push de l'image Docker locale vers notre Docker Hub. Nous avons donc notre image docker présent sur notre Docker Hub.

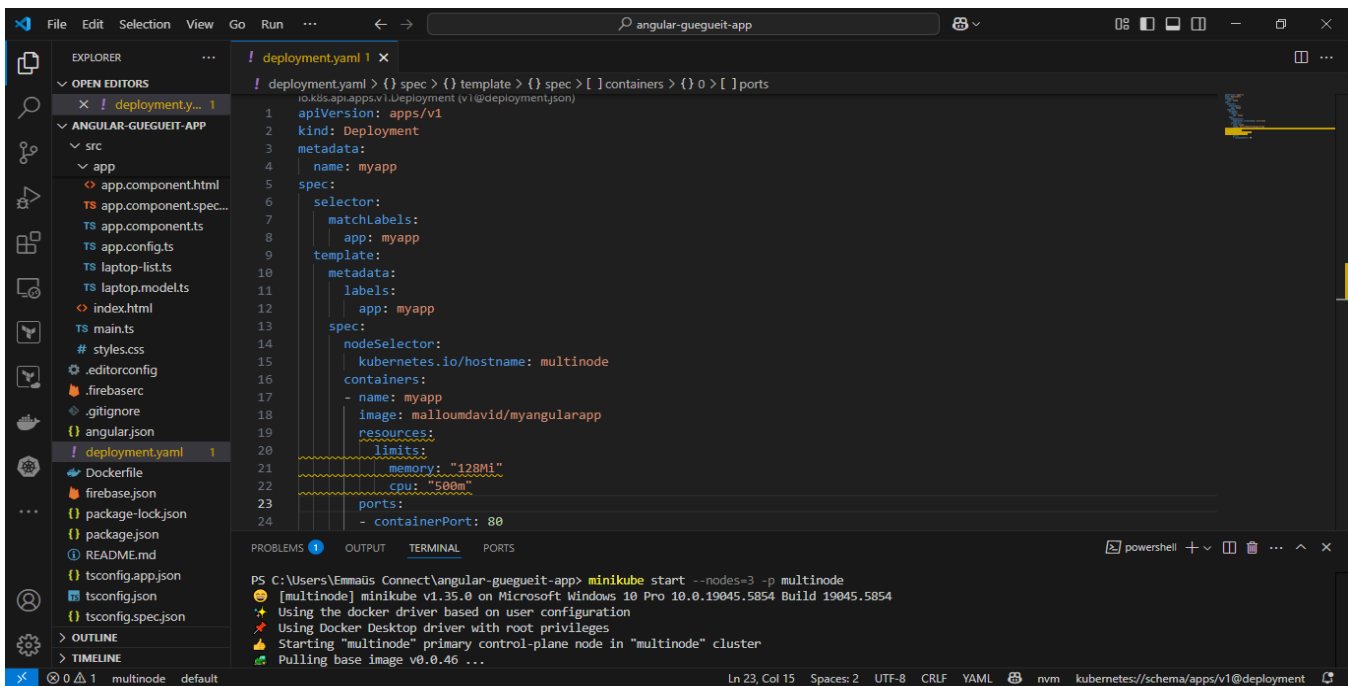


Nous avons donc notre image docker présent sur notre Docker Hub.



• Etape 5 :

Ensuite, nous allons modifier le fichier deployment.yaml et taper la commande "minikube start - -nodes=3 -p multinode" qui nous permet de démarrer un cluster Kubernetes local avec Minikube, en spécifiant les 3 nœuds dans le cluster (1 master + 2 workers) et un profil nommé «multinode».



- **Etape 6 :**

Et enfin, nous allons taper la commande "kubectl get nodes" qui nous permet lister tous les nœuds d'un cluster Kubernetes, avec des informations de base sur chacun d'eux. Nous avons donc les 3 nœuds présents sur notre Docker Desktop.

```

Starting "multinode-m03" worker node in "multinode" cluster
Pulling base image v0.0.46 ...
Creating docker container (CPUs=2, Memory=2200MB) ...
Found network options:
Pulling base image v0.0.46 ...
Creating docker container (CPUs=2, Memory=2200MB) ...
Found network options:
Found network options:
• NO_PROXY=192.168.76.2,192.168.76.3
• NO_PROXY=192.168.76.2,192.168.76.3
Falling to connect to https://registry.k8s.io/ from inside the minikube container
To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
• NO_PROXY=192.168.76.2,192.168.76.3
Falling to connect to https://registry.k8s.io/ from inside the minikube container
To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
• env NO_PROXY=192.168.76.2
• env NO_PROXY=192.168.76.2,192.168.76.3
Verifying Kubernetes components...
Verifying Kubernetes components...
Verifying Kubernetes components...
Verifying Kubernetes components...
! C:\Program Files\Docker\Docker\resources\bin\kubectl.exe is version 1.30.2, which may have incompatibilities with Kubernetes 1.32.0.
* Want kubectl v1.32.0? Try 'minikube kubectl -- get pods -A'
Done! kubectl is now configured to use "multinode" cluster and "default" namespace by default
PS C:\Users\Emmaüs Connect\angular-guegueit-app> kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
multinode     Ready    control-plane   20m   v1.32.0
multinode-m02 Ready    <none>        16m   v1.32.0
multinode-m03 Ready    <none>        12m   v1.32.0
PS C:\Users\Emmaüs Connect\angular-guegueit-app>

```

Nous avons donc les 3 nœuds présents sur notre Docker Desktop.

Container CPU usage: 66.70% / 400% (4 CPUs available)

Container memory usage: 1.16GB / 3.69GB

Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	Actions
sleepy_bhabha	da7203900908	my-angular	4201:4200	0%	0B / 0B	[Play] [Stop] [Trash]
multinode	bb762f38311e	k8s-minikul	53040:22 530...	19.73%	716.2MB / 2.15GI	[Play] [Stop] [Trash]
multinode-m02	27b34d0e0774	k8s-minikul	53175:22 531...	3.51%	223.3MB / 2.15GI	[Play] [Stop] [Trash]
multinode-m03	db8987465e44	k8s-minikul	53287:22 532...	4.6%	250.2MB / 2.15GI	[Play] [Stop] [Trash]

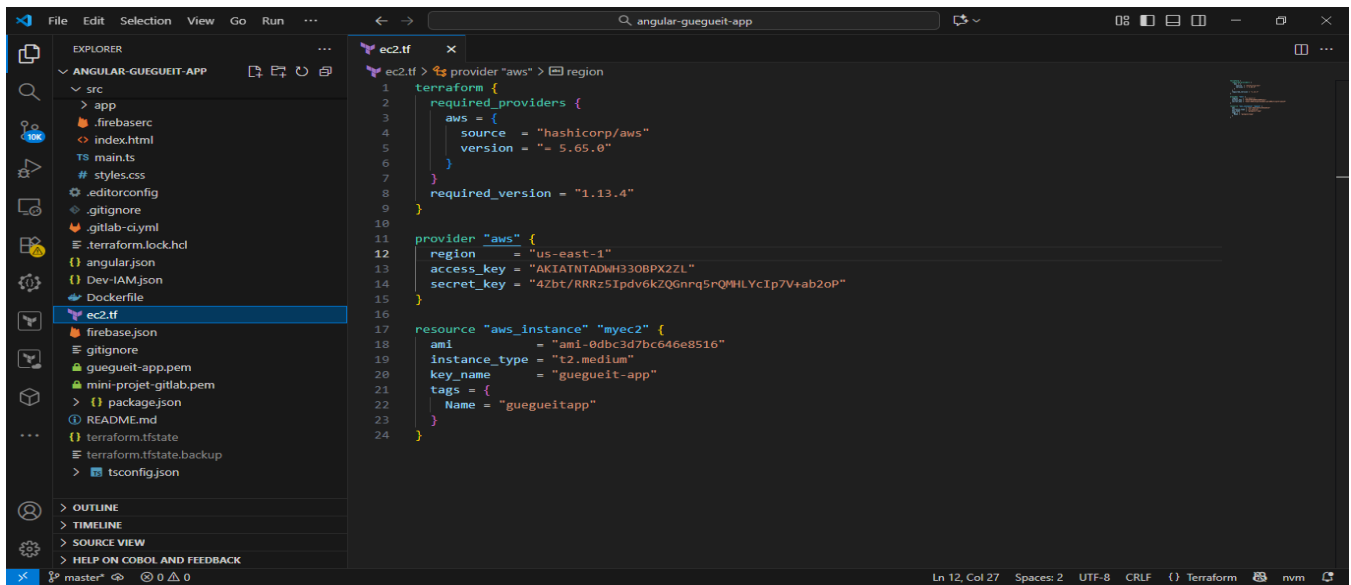
Showing 4 items

Engine running | Kubernetes running RAM 3.84 GB CPU 36.29% Disk --- GB avail. of --- GB | BETA Terminal New version available 8

🚀 Création des instances ec2 avec Terraform

- Etape 1 :

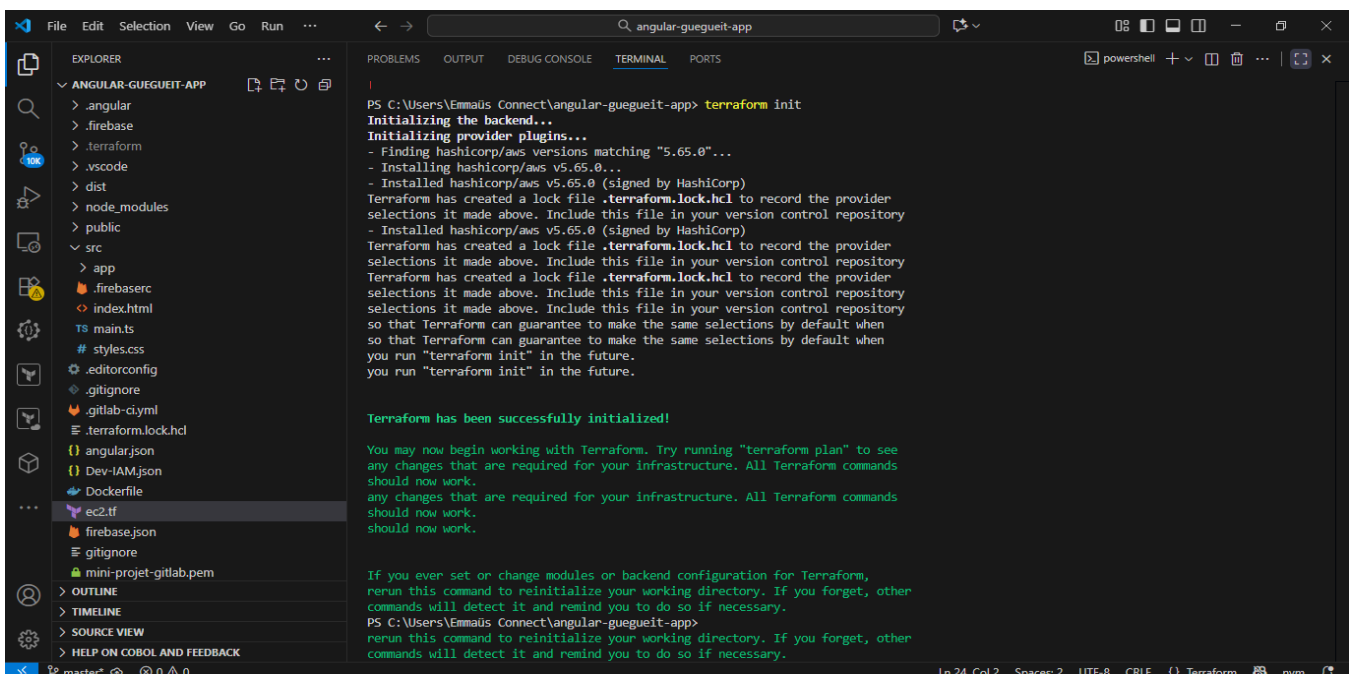
Nous allons créer et configurer le fichier ec2.tf



```
1 provider "aws" {
2   region
3 }
4 terraform {
5   required_providers {
6     aws = {
7       source  = "hashicorp/aws"
8       version = "~> 5.65.0"
9     }
10  }
11  required_version = "1.13.4"
12 }
13
14 provider "aws" {
15   region     = "us-east-1"
16   access_key = "AKIATNTADWH330BPX2ZL"
17   secret_key = "4Zbt/RRRz5Ipdv6kZQnq5rQMHLyCp7V+ab2oP"
18 }
19
20 resource "aws_instance" "myec2" {
21   ami           = "ami-0d8c3d7bc646e8516"
22   instance_type = "t2.medium"
23   key_name      = "guegueit-app"
24   tags = {
25     Name = "guegueitapp"
26   }
27 }
```

- Etape 2 :

Ensuite, au niveau du terminal, nous allons taper la commande « terraform init » qui nous permet d’initialiser le répertoire de travail contenant les fichiers de configuration Terraform (.tf).



```
PS C:\Users\Emmaüs Connect\angular-guegueit-app> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.65.0"...
- Installing hashicorp/aws v5.65.0...
- Installed hashicorp/aws v5.65.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
- Installed hashicorp/aws v5.65.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.
Terraform has been successfully initialized!

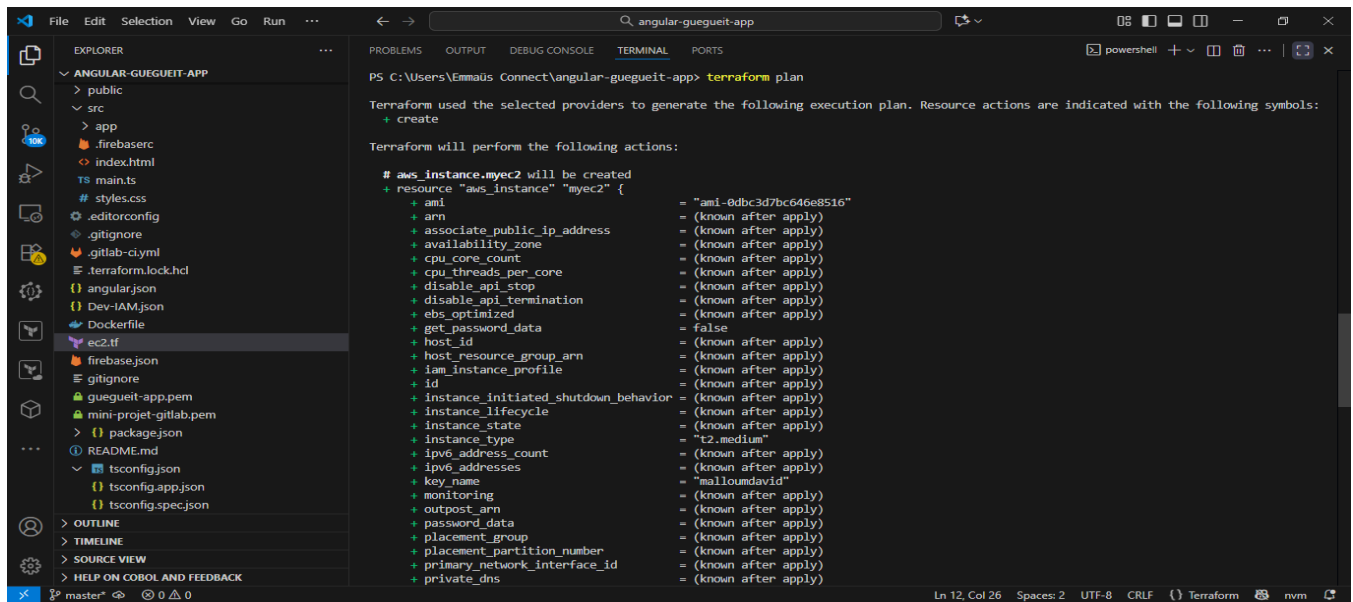
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\Emmaüs Connect\angular-guegueit-app>
```

- Etape 3 :

Ensuite, nous allons taper la commande « terraform plan » qui nous permet de prévisualiser les changements que Terraform va appliquer à notre infrastructure, sans les exécuter réellement.



The screenshot shows the Visual Studio Code interface with the Explorer view on the left displaying the project structure. The Terminal view on the right shows the execution of the 'terraform plan' command. The output indicates that Terraform will create an AWS instance named 'myec2' with various configurations. The status bar at the bottom shows 'Ln 12, Col 26'.

```
PS C:\Users\Emmaüs Connect\angular-guegueit-app> terraform plan

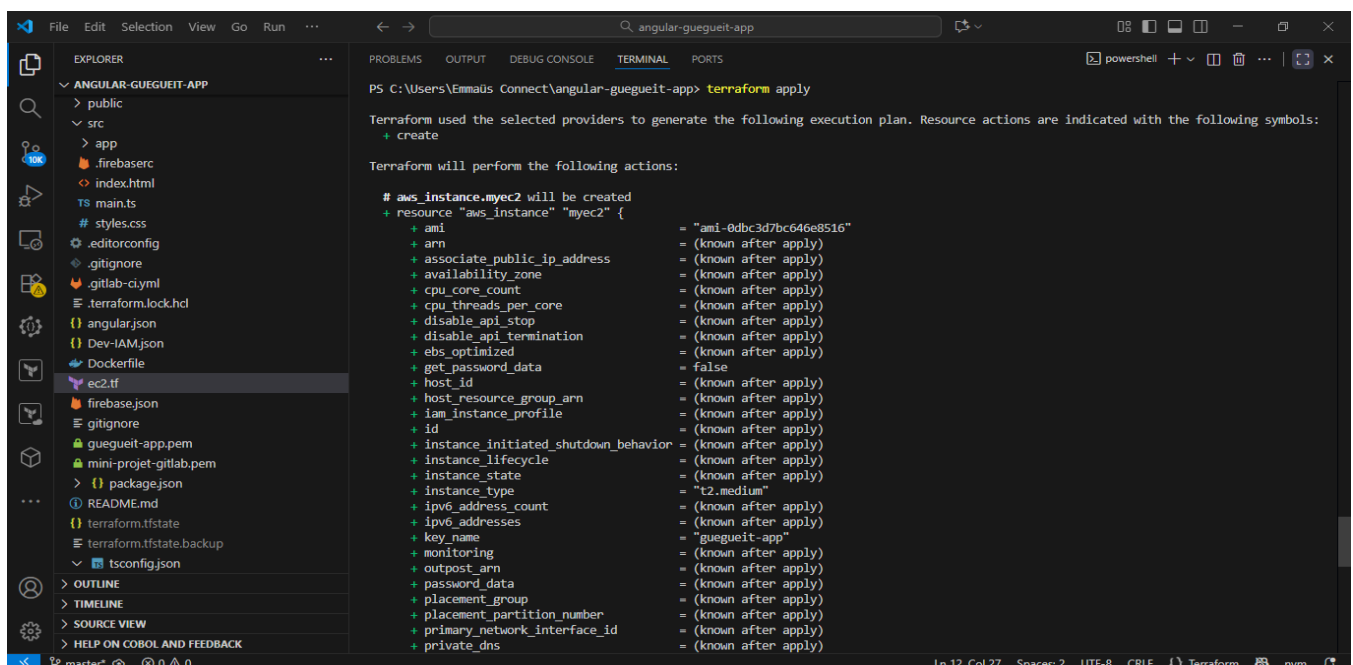
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.myec2 will be created
+ resource "aws_instance" "myec2" {
  + ami                  = "ami-0dbc3d7bc646e8516"
  + arn                  = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone     = (known after apply)
  + cpu_core_count       = (known after apply)
  + cpu_threads_per_core = (known after apply)
  + disable_api_stop      = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized         = (known after apply)
  + get_password_data     = false
  + host_id               = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile  = (known after apply)
  + id                    = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle    = (known after apply)
  + instance_state        = (known after apply)
  + instance_type         = "t2.medium"
  + ipv6_address_count    = (known after apply)
  + ipv6_addresses       = (known after apply)
  + key_name              = "malloumdavid"
  + monitoring            = (known after apply)
  + outpost_arn           = (known after apply)
  + password_data         = (known after apply)
  + placement_group       = (known after apply)
  + placement_partition_number = (known after apply)
  + primary_network_interface_id = (known after apply)
  + private_dns           = (known after apply)
}
```

- Etape 3 :

Ensuite, nous allons taper la commande « terraform apply » qui nous permet d'exécuter réellement les changements sur notre infrastructure.



The screenshot shows the Visual Studio Code interface with the Explorer view on the left displaying the project structure. The Terminal view on the right shows the execution of the 'terraform apply' command. The output indicates that Terraform will create an AWS instance named 'myec2' with various configurations. The status bar at the bottom shows 'Ln 12, Col 27'.

```
PS C:\Users\Emmaüs Connect\angular-guegueit-app> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.myec2 will be created
+ resource "aws_instance" "myec2" {
  + ami                  = "ami-0dbc3d7bc646e8516"
  + arn                  = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone     = (known after apply)
  + cpu_core_count       = (known after apply)
  + cpu_threads_per_core = (known after apply)
  + disable_api_stop      = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized         = (known after apply)
  + get_password_data     = false
  + host_id               = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile  = (known after apply)
  + id                    = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle    = (known after apply)
  + instance_state        = (known after apply)
  + instance_type         = "t2.medium"
  + ipv6_address_count    = (known after apply)
  + ipv6_addresses       = (known after apply)
  + key_name              = "guegueit-app"
  + monitoring            = (known after apply)
  + outpost_arn           = (known after apply)
  + password_data         = (known after apply)
  + placement_group       = (known after apply)
  + placement_partition_number = (known after apply)
  + primary_network_interface_id = (known after apply)
  + private_dns           = (known after apply)
}
```

```
File Edit Selection View Go Run ... angular-guegueit-app
EXPLORER
ANGULAR-GUEGUEIT-APP
  public
  src
    app
      .firebaserc
      index.html
    TS mains
    # styles.css
    .editorconfig
    .gitignore
    .gitlab-ci.yml
    .terraform.lock.hcl
    angular.json
    Dev-IAM.json
    Dockerfile
    ec2.tf
    firebase.json
    gitignore
    guegueit-app.pem
    mini-projet-gitlab.pem
    package.json
    README.md
    terraform.tfstate
    terraform.tfstate.backup
    tsconfig.json
  OUTLINE
  TIMELINE
  SOURCE VIEW
  HELP ON COBOL AND FEEDBACK

TERMINAL
+ capacity_reservation_specification (known after apply)
+ cpu_options (known after apply)
+ ebs_block_device (known after apply)
+ enclave_options (known after apply)
+ ephemeral_block_device (known after apply)
+ instance_market_options (known after apply)
+ maintenance_options (known after apply)
+ metadata_options (known after apply)
+ network_interface (known after apply)
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}

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

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

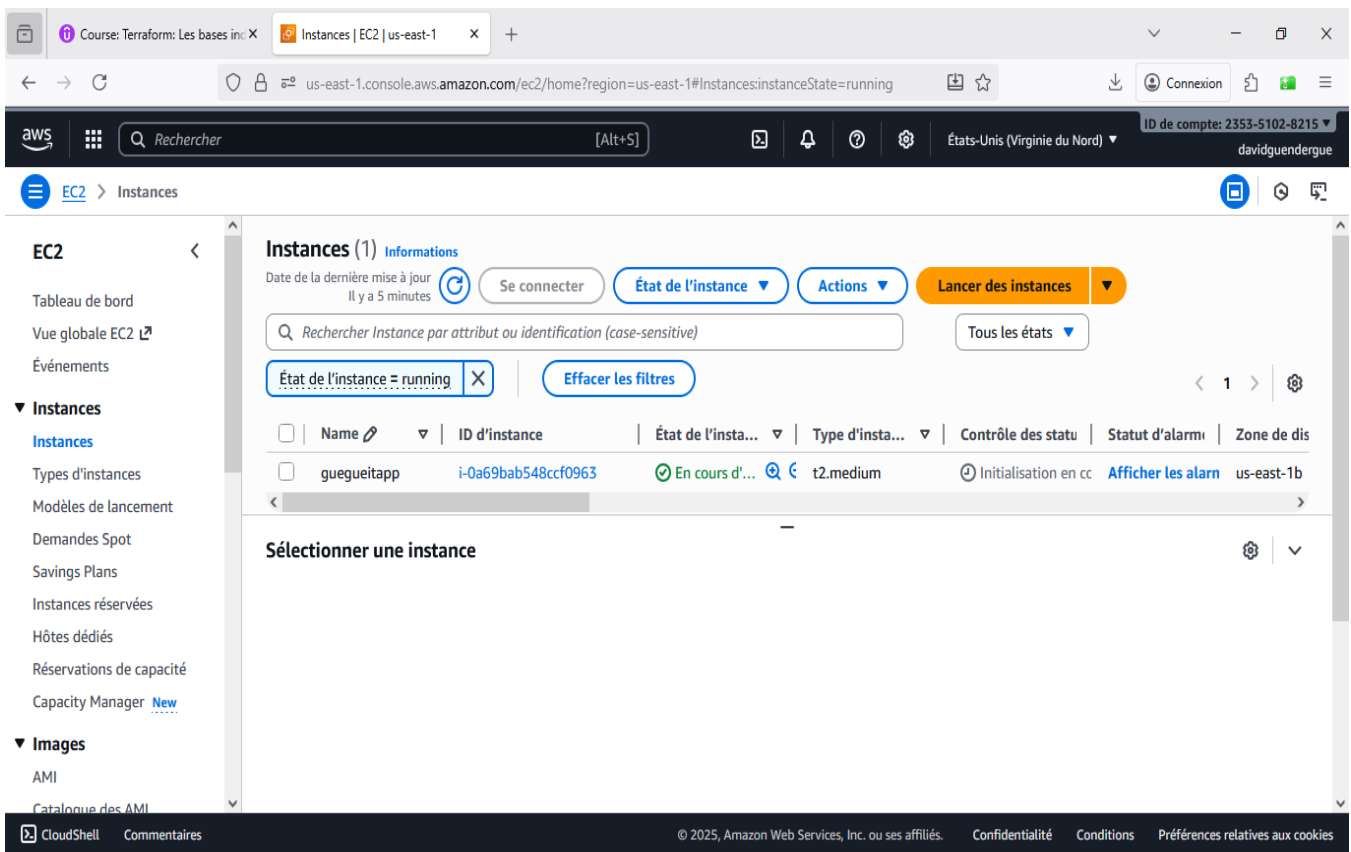
Enter a value: yes

aws_instance.myc2: Creating...
aws_instance.myc2: Still creating... [00m10s elapsed]
aws_instance.myc2: Creation complete after 15s [id=i-0a69bab548ccf0963]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Emmaüs Connect\angular-guegueit-app>
```

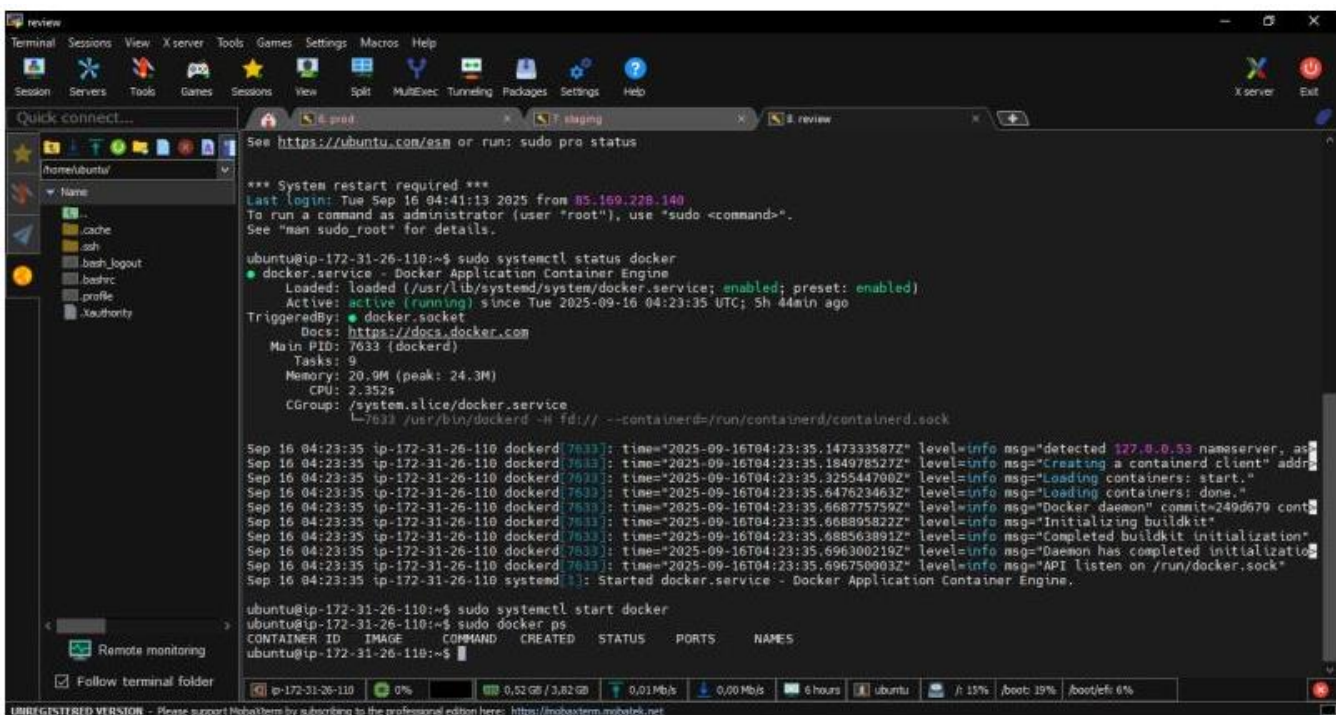
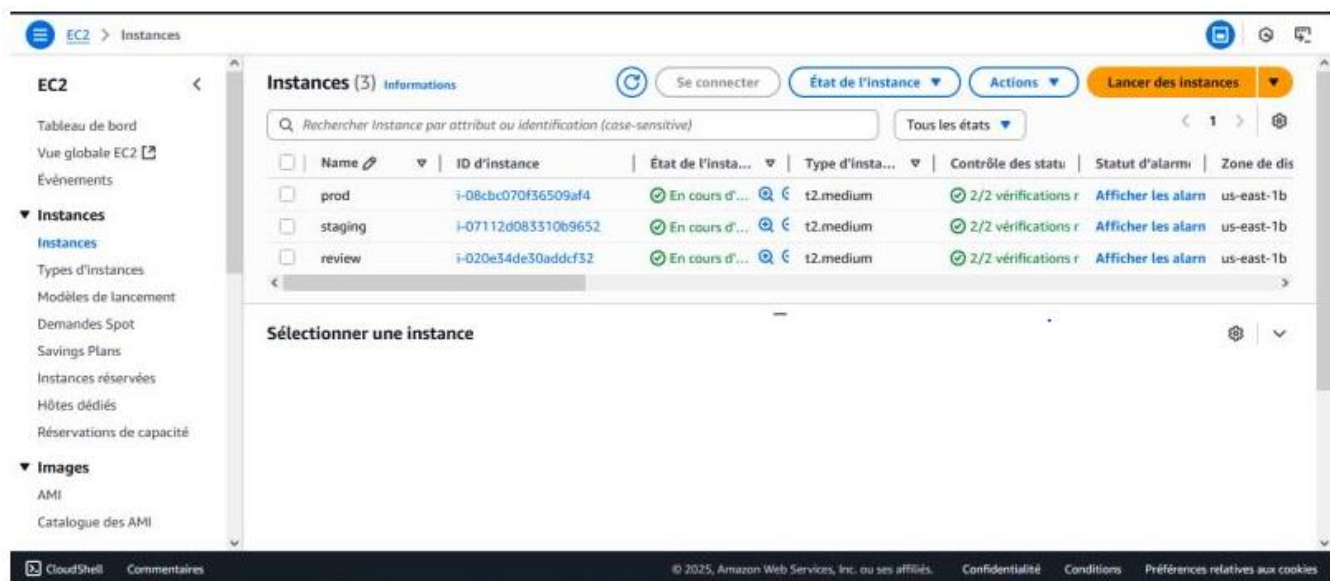
• Etape 4 :

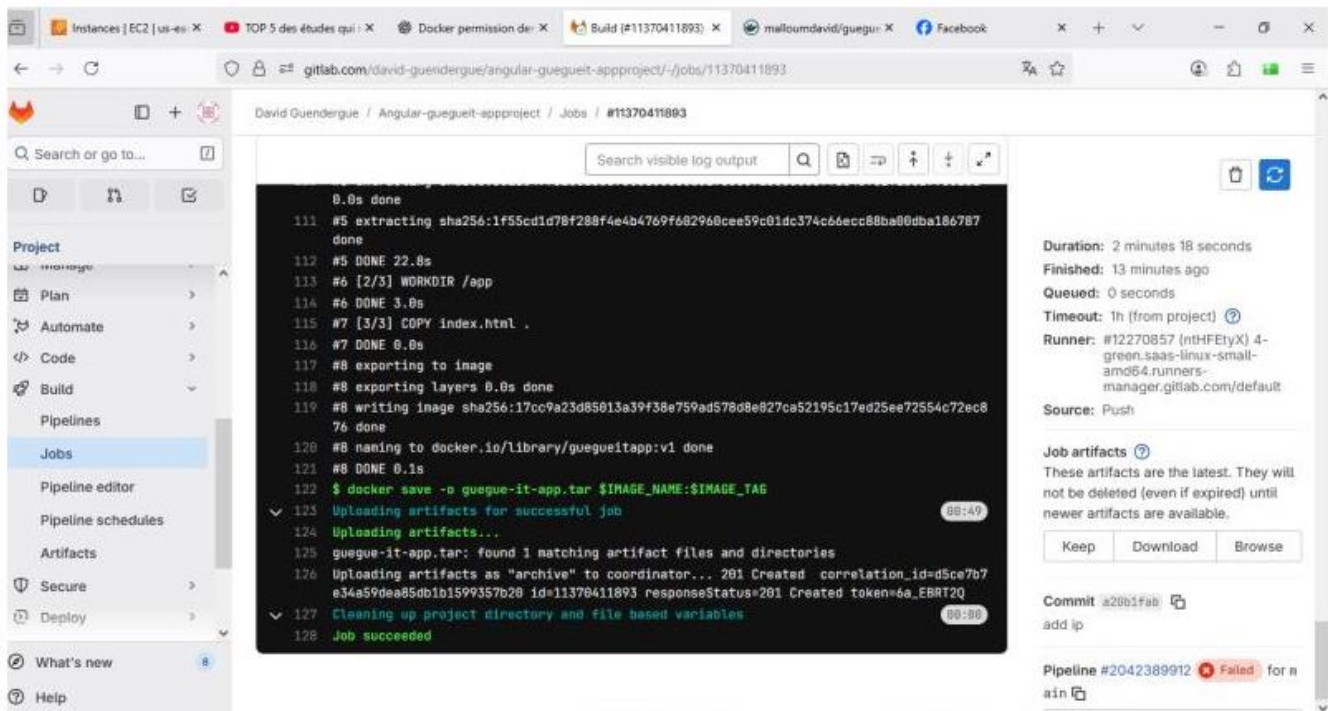
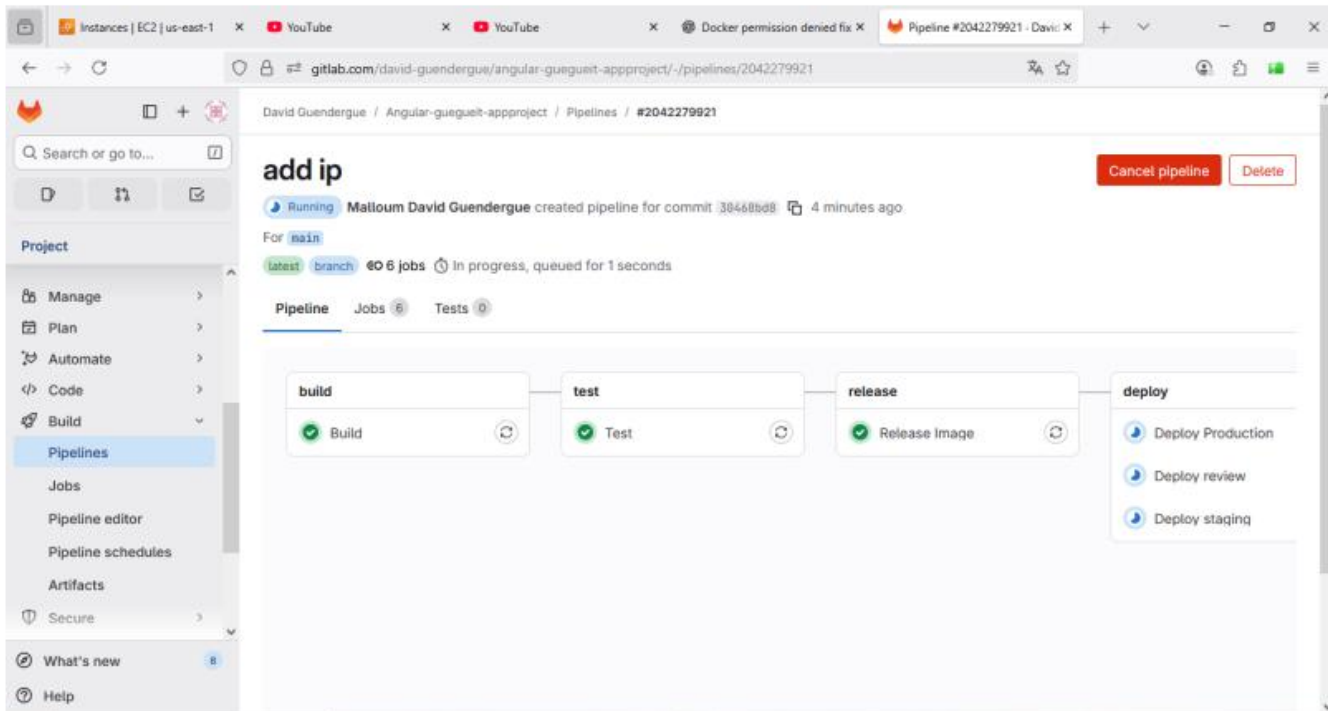
Et enfin, nous allons pouvoir apercevoir notre instance ec2 créée sur la console de notre compte AWS.



🚦 Mise en place d'un pipeline CI/CD avec GitLab

Nous avons tout d'abord créé 3 instances ec2 (prod, staging et review) sur AWS, ensuite nous avons visualiser ces instances ec2 sur MobaXterm, ensuite nous avons utilisé GitLab pour transformer les configurations du DockerFile obtenu en langage shell, effectuer le push de notre code sur notre GitHub et sur notre GitLab, effectuer des configurations de sécurité avec notre Docker Hub et sur base64 pour la clé publique et enfin nous avons modifié l'éditeur de pipeline en fonction des résultats obtenus afin de cliquer sur le bouton «Commit changes».





Instances | EC2 | us-east-1 | TOP 5 des études qui | Docker permission de | Test (#11370411905) | malloundavid/guegu: | Facebook

gitlab.com/david-guendergue/angular-gueguet-appproject/-/jobs/11370411905

David Guendergue / Angular-gueguet-appproject / Jobs / #11370411905

Search or go to...

Project

- Plan
- Automate
- Code
- Build
- Pipelines
- Jobs
- Pipeline editor
- Pipeline schedules
- Artifacts
- Secure
- Deploy

What's new

Help

Search visible log output

```
26 Downloading artifacts for Build (11370411893)...
27 Downloading artifacts from coordinator... ok correlation_id= host-storage.googleapis.com id=11370411893 responseStatus=200 OK token=6a_1KYk1r
28 Executing "step_script" stage of the job script
29 Using effective pull policy of [always] for container docker:latest
30 Using docker image sha256:14108d5b9ccb9bf761088a8576a1357c393dd7ba8d85cae02cbe797b3c59e69e for docker:latest with digest docker@sha256:831644212c5dd0b3362b5855c87b980ea39a83c9e9adcef2ce03ecad99e737a ...
31 $ apk add --no-cache curl
32 fetch https://dl-cdn.alpinelinux.org/alpine/v3.22/main/x86_64/APKINDEX.tar.gz
33 fetch https://dl-cdn.alpinelinux.org/alpine/v3.22/community/x86_64/APKINDEX.tar.gz
34 (1/1) Installing curl (8.14.1-r1)
35 Executing busybox-1.37.0-r18.trigger
36 OK: 41 MiB in 76 packages
37 $ docker load -i gueguet-app.tar
38 Loaded image: gueguetapp:v1
39 $ docker run --rm -dp $HOST_PORT:$CONTAINER_PORT --name $IMAGE_NAME $IMAGE_NAME:$IMAGE_TAG
40 aaafb18fc627d3c8408a294e8a98575f713b9c951a7e6d2a998ba1caf42c61b0
41 $ sleep 5
42 Cleaning up project directory and file based variables
43 Job succeeded
```

Duration: 1 minute 30 seconds
Finished: 12 minutes ago
Queued: 0 seconds
Timeout: 1h (from project)
Runner: #12270848 (ns46NMmJ) 2-green.saas-linux-small-amd64.runners-manager.gitlab.com/default
Source: Push
Commit a28b1fab
add ip
Pipeline #2042389912 Failed for main
test
Related jobs
Test

Instances | EC2 | us-east-1 | TOP 5 des études qui | Docker permission de | Release Image (#11370411917) | malloundavid/guegu: | Facebook

gitlab.com/david-guendergue/angular-gueguet-appproject/-/jobs/11370411917

David Guendergue / Angular-gueguet-appproject / Jobs / #11370411917

Search or go to...

Project

- Plan
- Automate
- Code
- Build
- Pipelines
- Jobs
- Pipeline editor
- Pipeline schedules
- Artifacts
- Secure
- Deploy

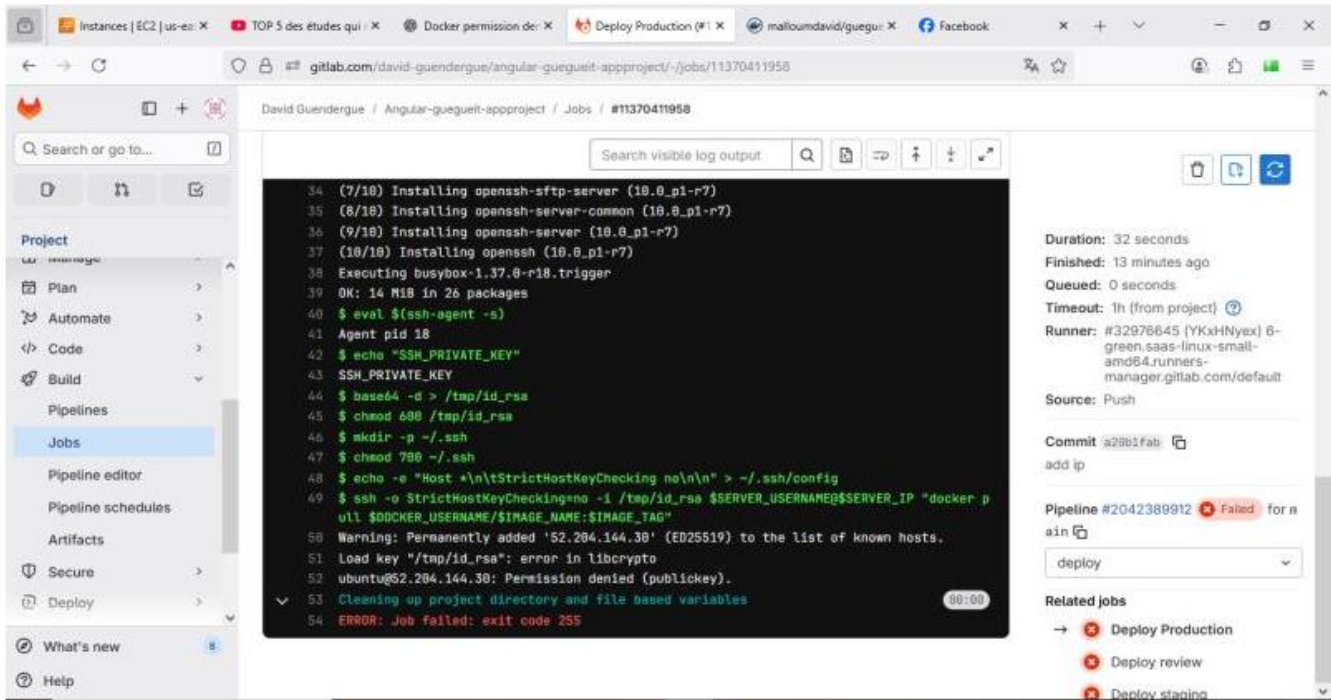
What's new

Help

Search visible log output

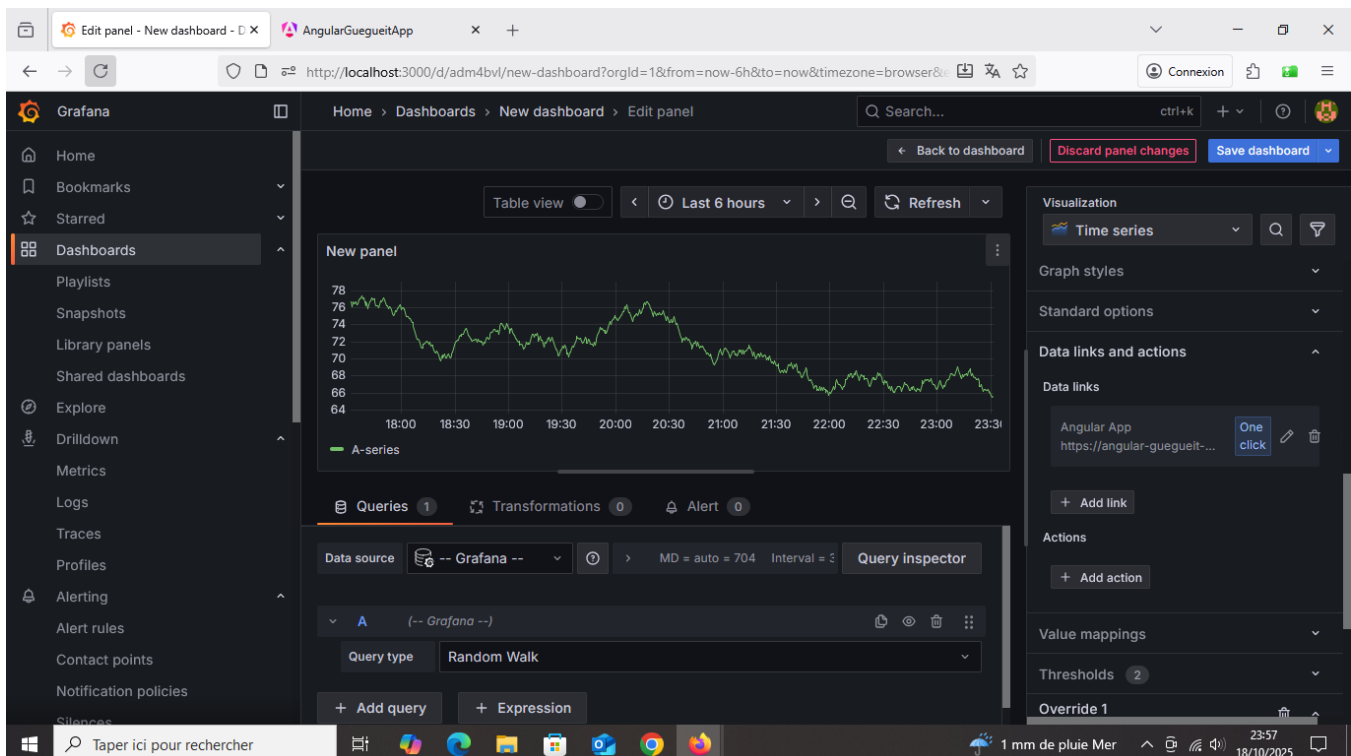
```
48 1e167435abc: Preparing
49 df36a4427956: Preparing
50 88466a243658: Preparing
51 eee2af66d91c: Waiting
52 29435ff34d4a: Waiting
53 1e167435abc: Waiting
54 df36a4427956: Waiting
55 88466a243658: Waiting
56 b94df955776a: Pushed
57 8348fdb2114: Pushed
58 1c4b33828f88: Pushed
59 eee2af66d91c: Pushed
60 954ead644fd7: Pushed
61 be4db13b04fd: Pushed
62 df36a4427956: Pushed
63 88466a243658: Pushed
64 1e167435abc: Pushed
65 29435ff34d4a: Pushed
66 v1: digest: sha256:f468fbd36cb0b6fdbcb4fe491863bd0f772cc83500278b6c62d19b3a5ce2456eaf s1ze: 2417
67 Cleaning up project directory and file based variables
68 Job succeeded
```

Duration: 1 minute 58 seconds
Finished: 12 minutes ago
Queued: 0 seconds
Timeout: 1h (from project)
Runner: #12270848 (ns46NMmJ) 2-green.saas-linux-small-amd64.runners-manager.gitlab.com/default
Source: Push
Commit a28b1fab
add ip
Pipeline #2042389912 Failed for main
release
Related jobs
Release Image



E- Observabilité & opérations (dashboards, alertes, runbooks) :

Nous avons réalisé nos travaux d'observabilité et d'opérations de notre application sur Grafana comme le montre la capture d'écran ci-dessous de notre graphe.



F- Tests & résultats (méthode, outils, mesures, limites) :

Pour les tests et les résultats, nous avons :

- ✚ La méthode utilisée est la culture DevOps.
- ✚ Les outils utilisés sont Angular, AWS, Docker, Minikube, GitLab, Terraform, Firebase, Grafana et AWS Pricing Calculator.
- ✚ Les mesures utilisées pour la création des instances ec2 sont par exemple le type d'instance « t2.medium » et la région « us-east-1 ».
- ✚ La limite principale que nous avons rencontrée est au niveau des 3 stages de Deploy sur le pipeline CI/CD de GitLab qui n'a pas pu se réaliser avec succès.

G- Coûts & optimisation (FinOps) :

- ✚ Nous utilisons 3 instances ec2 t2.medium en mode On-Demand (24/7) et nous utilisons « AWS Pricing Calculator » pour effectuer nos calculs. La région choisie est : US East (N. Virginia) (us-east-1)

- En calculant le coût mensuel total en On-Demand, nous avons :

Coût mensuel total = 33,87 USD/mois

- En estimant le coût mensuel avec des Reserved Instances, nous avons :

Coût mensuel avec des Reserved Instances = 20,95 USD/mois

- En estimant le coût mensuel avec un Compute Saving Plan, nous savons :

Coût mensuel avec un Compute Saving Plan = 24,16 USD/mois

- En comparant les 3 options dans un tableau (Type, Coût mensuel, Économie vs OnDemand), nous pouvons conclure que la meilleure stratégie FinOps est la stratégie des Reserved Instances parce qu'elle est la stratégie ayant le coût le moins cher et le plus économique parmi les 3 options disponibles.

CONCLUSION (Risques, limites, perspectives)

En définitive, nous pouvons affirmer que le choix du Framework Angular pour développer l'application était la meilleure option parce qu'il est basé sur le langage JavaScript qu'on peut utiliser en Front end comme en Back end, et il nous permet également de consommer moins de ressources en production par rapport à une même application écrite en langage PHP par exemple. Un logiciel fiable, ce n'est pas juste du code qui tourne sans erreur. C'est un produit qui protège les données, contrôle qui peut y accéder et détecte les incidents avant qu'ils ne dégénèrent. Les infrastructures informatiques mal gérées génèrent des problématiques sérieuses qui impactent directement la performance et la sécurité d'une entreprise. En effet, des pannes système, des performances limitées, des pertes de données, ainsi que des lenteurs de connexion peuvent survenir. L'hétérogénéité des technologies, la diversité des équipements (serveurs, réseaux, systèmes de stockage), ainsi que le manque de compétences techniques peuvent créer une surcharge de travail difficile à gérer. Au fil du temps, les coûts liés à la maintenance et à la gestion des infrastructures deviennent de plus en plus élevés. Dans ce contexte, confier cette gestion à un prestataire spécialisé en gestion d'infrastructures IT s'avère être une solution stratégique et plus rentable.

REFERENCES & ANNEXES

- + Jacques Printz (préf. Yves Caseau), *Architecture logicielle : concevoir des applications simples, sûres et adaptables*, Paris, Dunod, 2012, 3e éd., 512 p.
- + Joëlle Delacroix et Alain Cazes, *Architecture des machines et des systèmes informatiques*, Paris, Dunod, 2018, 6^e éd., 560 p.
- + <https://angular.dev/>
- + <https://aws.amazon.com/fr/>
- + <https://www.docker.com/>
- + <https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+download>
- + <https://about.gitlab.com/>
- + https://www.hashicorp.com/fr/products/terraform?utm_source=google&utm_channel_bucket=paid&utm_medium=sem&utm_campaign=Speed_EMEA_ENG_BOFU_Practitioner_SEM_Brand-Terraform_ILM&utm_content=terraform%20cloud-144080651246-645564370290&utm_offer=signup&gad_source=1&gad_campaignid=19592887117&gclid=CjwKCAjwjffHBhBuEiwAKMb8pEi-pSrJ-Lulw-G1uptkOQngLT_87yOrzwwkeO462orfwPII-BxTIBoCO9MQAvD_BwE
- + <https://firebase.google.com/>
- + <https://grafana.com/>
- + <https://calculator.aws/#/>