

## SOMMAIRE

<b>RESUME EXECUTIF</b>	<b>1</b>
<b>INTRODUCTION (Contexte, hypothèses, exigences)</b>	<b>2</b>
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
<b>CONCLUSION (Risques, limites, perspectives)</b>	<b>19</b>
<b>REFERENCES &amp; ANNEXES</b>	<b>20</b>

## **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'incrémentation 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**

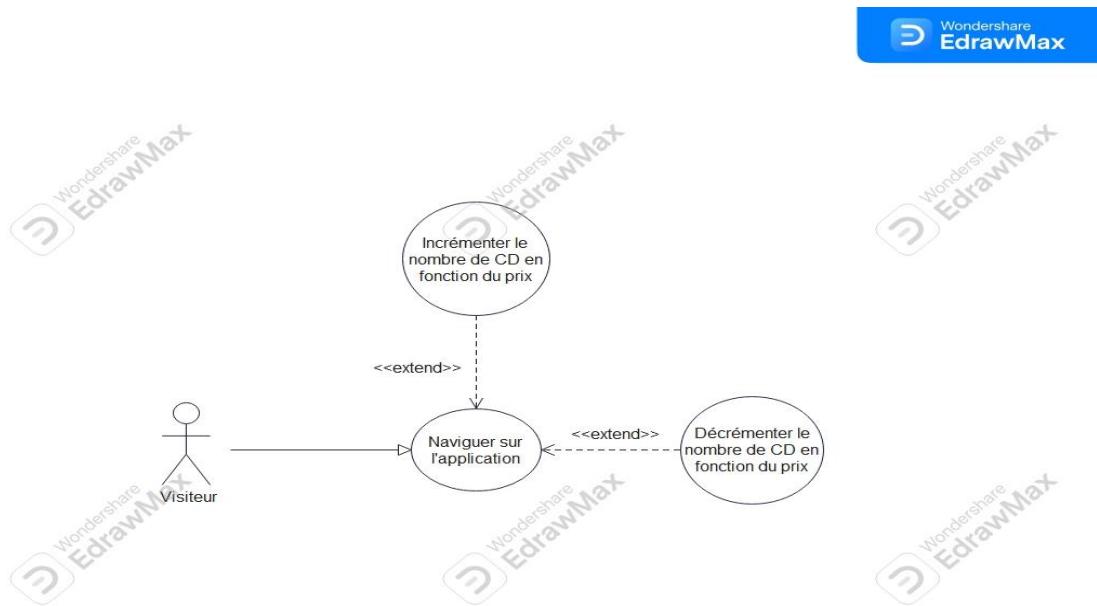
Solution technique : La solution technique que nous privilégions pour le développement de l'application est l'utilisation du Framework Angular de Google à travers le logiciel open source Visual Studio Code qui est un environnement de développement intégré ou IDE. Cas du nom de domaine et de l'hébergement : 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 (Unified Modeling Language 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

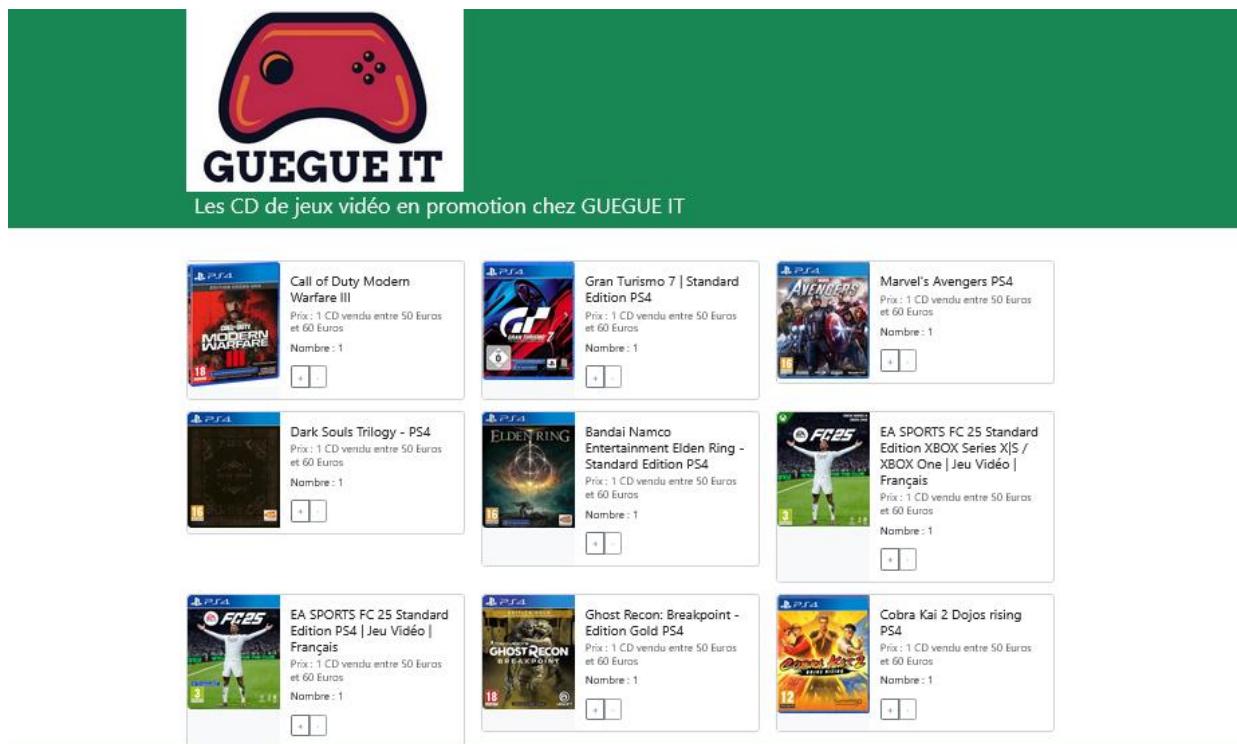
- Titre : Diagramme de cas d'utilisation Visiteur.
- Résumé : Nous avons la représentation du visiteur qui peut naviguer sur l'application, en faisant l'incrémentation et la décrémentation du nombre de CD en fonction du prix.
- Acteur : Visiteur.
- Motivation : L'acteur veut naviguer sur l'application.
- 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://angulargueueit-app.web.app/>



The screenshot shows the homepage of the GUEUE IT website. At the top, there's a green header with a red game controller icon and the text "GUEUE IT" in white. Below the header, it says "Les CD de jeux vidéo en promotion chez GUEUE IT". The main content area displays a grid of 12 video game titles, each with a small image, the title, a price range, and a quantity selector (two input fields with plus and minus signs). The games listed are:

- Call of Duty Modern Warfare III
- Gran Turismo 7 | Standard Edition PS4
- Marvel's Avengers PS4
- Dark Souls Trilogy - PS4
- Elden Ring
- EA SPORTS FC 25 Standard Edition XBOX Series X|S / XBOX One | Jeu Vidéo | Français
- EA SPORTS FC 25 Standard Edition PS4 | Jeu Vidéo | Français
- Ghost Recon: Breakpoint - Edition Gold PS4
- Cobra Kai 2 Dojos rising PS4

## 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).

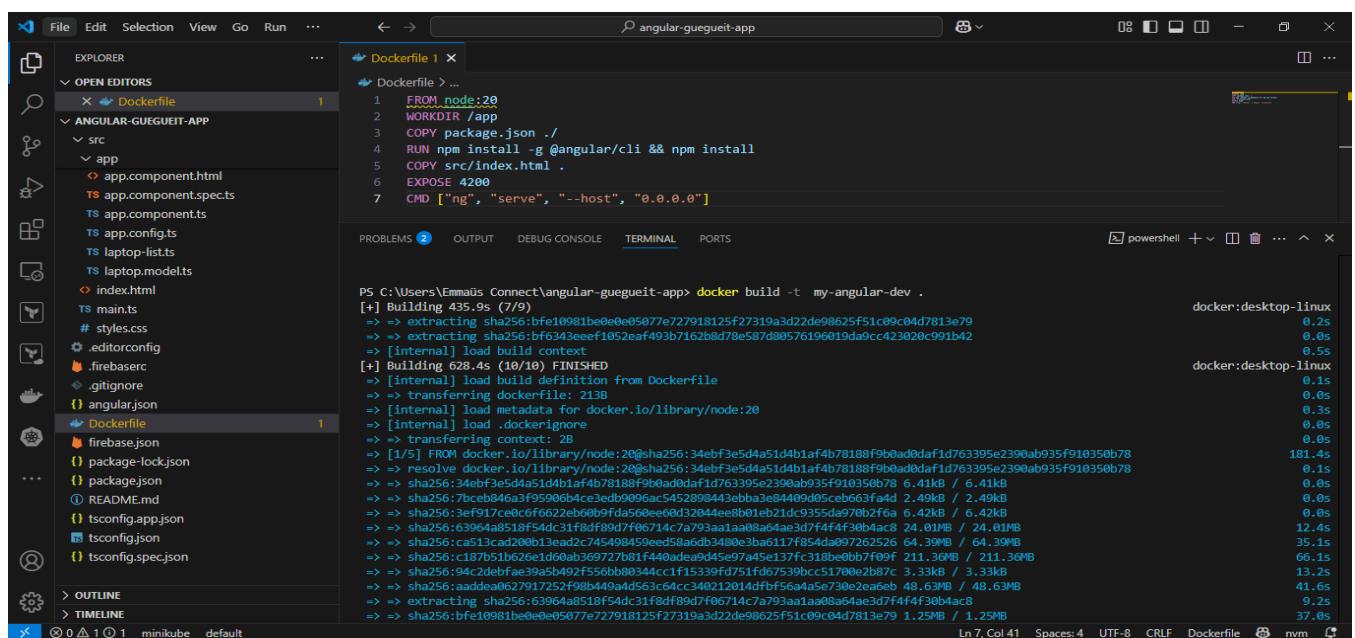
Nous avons le code json disponible sur le référentiel du code en archive .zip

## D- Déploiement (laC, 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 (.).



The screenshot shows the VS Code interface with the Dockerfile open in the editor. The terminal below shows the execution of the docker build command, which takes approximately 37.85 seconds to complete. The output includes details about extracting layers and building the image.

```
PS C:\Users\Emmaüs Connect\angular-gueueit-app> docker build -t my-angular-dev .
[+] Building 435.9s
--> extracting sha256:bfe10981be0e0e05077e27918125f27319a3d22de98625f51c09c04d7813e79
--> extracting sha256:bf6343eeef1052eaaf93b7162b8d78e587d805761966019da9cc423020c991b42
--> [internal] load build context
[+] Building 628.4s (10/10) FINISHED
--> [internal] load build definition from Dockerfile
--> transferring dockerfile: 213B
--> [internal] load metadata for docker.io/library/node:20
--> [internal] load .dockerignore
--> transferring context: 2B
--> [1/5] FROM docker.io/library/node:20@sha256:34ebf3e5d4a51d4b1af4b78188f9b0ad0daf1d763395e2390ab935f910350b78
--> > extracting sha256:34ebf3e5d4a51d4b1af4b78188f9b0ad0daf1d763395e2390ab935f910350b78
--> > extracting sha256:7bcb846a3f95906b4c5e2ed0b0095e5452899443cbb2ae84409d05ceb663faad2_49k / 6.41kB / 6.41kB
--> > sha256:34ebf3e5d4a51d4b1af4b78188f9b0ad0daf1d763395e2390ab935f910350b78
--> > sha256:7bcb846a3f95906b4c5e2ed0b0095e5452899443cbb2ae84409d05ceb663faad2_49k / 2.49kB / 2.49kB
--> > sha256:3ef917ce0cff6622eb60b9fd560e6b032944ee8b81cb21dc935fa970bf2fa6 6.42kB / 6.42kB
--> > sha256:639614a8518f51d31f8d789d7f66714c7a79aa1aa08a45fae3d7f4f4f38b4ac8 24.01MB / 24.01MB
--> > sha256:ca513cad200b13ead2c74598459eed5baed3480e3ba6117f8354da097262526 64.39MB / 64.39MB
--> > sha256:c187b5262e0db0ab369772881f40adea9d45e97445e137fc318be0bb7109f 211.36MB / 211.36MB
--> > sha256:94c2debfaf39a5d92f550bb8344c1f15339fd751fd67539bc51700e2b87c 3.33kB / 3.33kB
--> > sha256:aaddea0627917252f98b449a4d53c64cc340212814dfbf56a4a5e380e2a66b 48.63MB / 48.63MB
--> > extracting sha256:639614a8518f51d31f8d789d7f66714c7a79aa1aa08a44ea3d7f4f4f30b4ac8
--> > sha256:bfe10981be0e0e05077e27918125f27319a3d22de98625f51c09c04d7813e79 1.25MB / 1.25MB
```

#### • 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.

```

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

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Emmaüs Connect\angular-gueguoit-app> docker run -d -it -p 4201:4200 my-angular-dev  
da7203900908aa9fb84a84029e696f9df01450dbaa09e0d43de4ab2f1c16322  
PS C:\Users\Emmaüs Connect\angular-gueguoit-app>

Nous avons donc notre conteneur appelé «sleepy\_bhabha» présent sur notre Docker Desktop.

	Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	M Actions
<input type="checkbox"/>	sleepy_bhabha	da7203900908	my-angular	4201:4200	N/A	N/A	<input type="button"/> <input type="button"/> <input type="button"/>

Showing 1 item

- **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.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
        - name: myapp
          image: malloumdavid/myangularapp
          resources:
            limits:
              memory: "128Mi"
              cpu: "500m"
          ports:
            - containerPort: 4200

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

+ FullyQualifiedErrorId : CommandNotFoundException  
PS C:\Users\Emmaüs Connect\angular-gueguet-app> docker tag mon-express-app1 malloumdavid/myangularapp  
PS C:\Users\Emmaüs Connect\angular-gueguet-app> docker login  
Authenticating with existing credentials...  
Login Succeeded

- **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.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
        - name: myapp
          image: malloumdavid/myangularapp
          resources:
            limits:
              memory: "128Mi"
              cpu: "500m"
          ports:
            - containerPort: 4200

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Login Succeeded  
PS C:\Users\Emmaüs Connect\angular-gueguet-app> docker push malloumdavid/myangularapp  
Using default tag: latest  
The push refers to repository [docker.io/malloumdavid/myangularapp]  
243be1461f17: Preparing  
786becbcc922: Preparing  
7b098d8db809: Preparing  
a30a49ef86fa: Preparing

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

The screenshot shows the Docker Hub interface. On the left, there's a sidebar for the user 'malloumdavid' with options like 'Repositories', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main area shows the repository 'malloumdavid/myangularapp'. It has a 'General' tab selected, showing a single tag 'latest' which was pushed 7 minutes ago. There are tabs for 'Tags', 'Image Management (BETA)', 'Collaborators', 'Webhooks', and 'Settings'. To the right, there's a 'Docker commands' section with the command 'docker push malloumdavid/myangularapp :tagname'. Below that is a 'buildcloud' advertisement.

- **Etape 5 :**

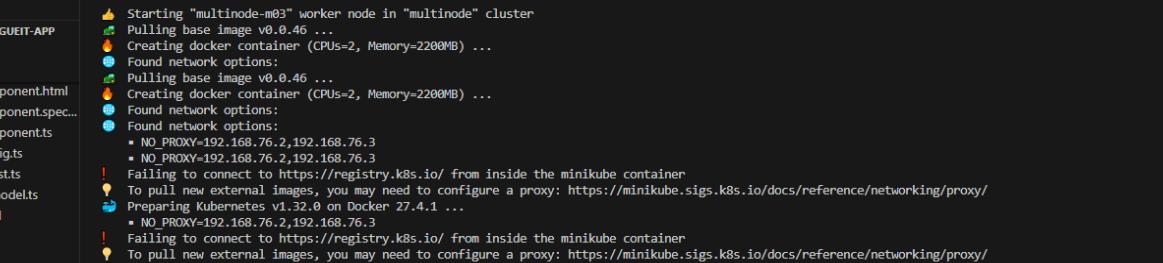
Ensute, nous allons modifier le fichier deployment.yaml et taper la commande "minikube start --nodes=3 -p multinode" qui nous permet de démarrer un cluster Kubernetes local avec Minikube, en spécifiant les 3 nœuds dans le cluster (1 master + 2 workers) et un profil nommé «multinode».

The screenshot shows the VS Code editor with the file 'deployment.yaml' open. The code defines a Kubernetes Deployment for an Angular application named 'myapp'. It specifies a Docker image of 'malloumdavid/myangularapp' and allocates 128Mi of memory and 500m of CPU. The deployment is targeted at pods with the label 'app: myapp'. The code is syntax-highlighted, and the terminal below shows the command 'minikube start --nodes=3 -p multinode' being run, indicating the cluster is starting up.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      nodeSelector:
        kubernetes.io/hostname: multinode
      containers:
        - name: myapp
          image: malloumdavid/myangularapp
          resources:
            limits:
              memory: "128Mi"
              cpu: "500m"
            ports:
              - containerPort: 80
```

#### • Etape 6 :

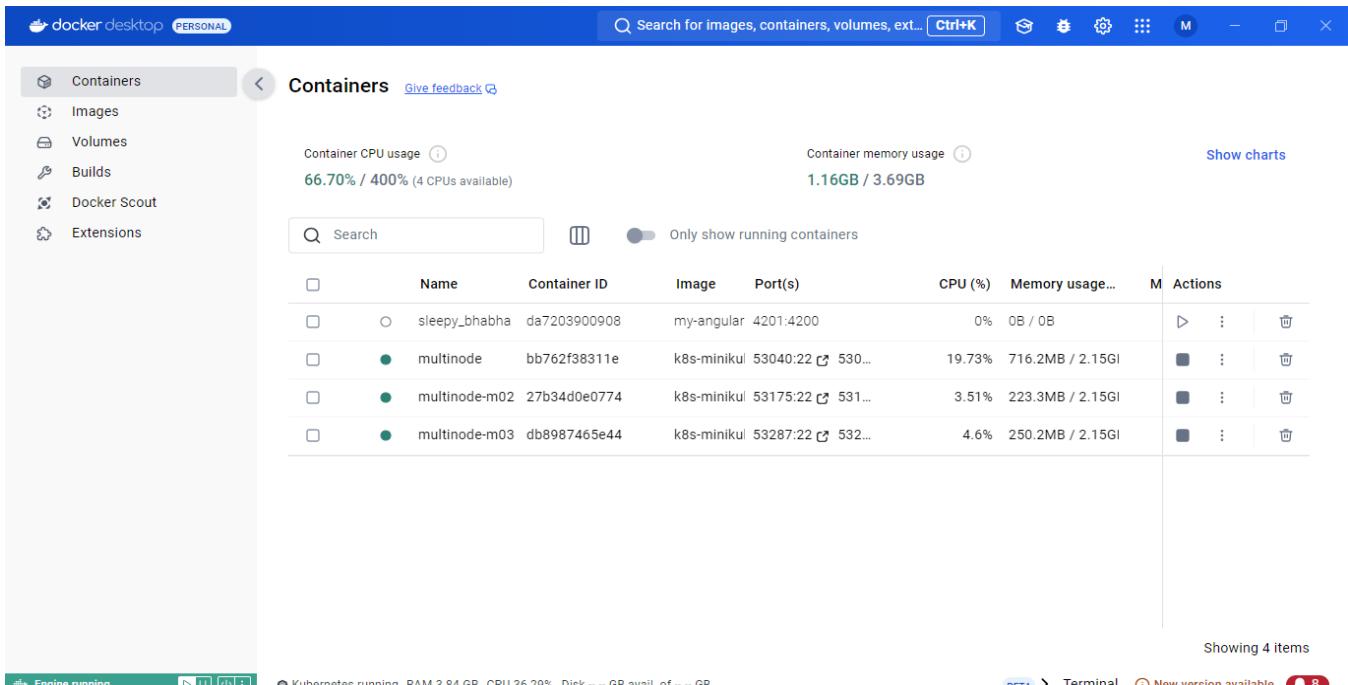
Et enfin, nous allons taper la commande "kubectl get nodes" qui nous permet lister tous les nœuds d'un cluster Kubernetes, avec des informations de base sur chacun d'eux. Nous avons donc les 3 nœuds présents sur notre Docker Desktop.



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it lists the project structure for "angular-guegueit-app". The "deployment.yaml" file is currently selected.
- Terminal:** The main area shows a PowerShell session with the following output:
  - 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
  - Failing 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
  - Failing 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
  - Failing 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
- Bottom Status Bar:** Shows icons for file status (modified), search, and navigation, along with the text "multinode default".

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



## Création des instances ec2 avec Terraform

- **Etape 1 :**

Nous allons créer et configurer le fichier ec2.tf

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure: ANGULAR-GUEUEIT-APP, src, and ec2.tf.
- Code Editor:** Displays the contents of the ec2.tf file:

```
provider "aws" {
  region = "us-east-1"
  access_key = "AKIAINTADNH33OBPX2ZL"
  secret_key = "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"
}

resource "aws_instance" "myec2" {
  ami           = "ami-0dbc3d7bc646e8516"
  instance_type = "t2.medium"
  key_name      = "gueueit-app"
  tags          = {
    Name = "gueueitapp"
  }
}
```

- Status Bar:** Shows the file is on master\*, with 0 changes, and the code is in UTF-8, CRLF format.

- **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).

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure: ANGULAR-GUEUEIT-APP, src, and ec2.tf.
- Terminal:** Displays the output of the terraform init command:

```
PS C:\Users\Emmaüs Connect\angular-gueueit-app> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.65.0"...
- Installing hashicorp/aws v5.65.0...
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.
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.

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-gueueit-app>
```

- Status Bar:** Shows the file is on master\*, with 0 changes, and the code is in UTF-8, CRLF format.

- **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.

```

File Edit Selection View Go Run ... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Emmaus Connect\angular-gueguet-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"
    ami_id                            = (known after apply)
    arm                               = (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               = (known after apply)
    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                          = "malloumavid"
    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)
}
Ln 12, Col 26 Spaces: 2 UTF-8 CRLF { } Terraform nvm

```

- **Etape 3 :**

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

```

File Edit Selection View Go Run ... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Emmaus Connect\angular-gueguet-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"
    ami_id                            = (known after apply)
    arm                               = (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               = (known after apply)
    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                          = "gueguet-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)
}
Ln 12, Col 27 Spaces: 2 UTF-8 CRLF { } Terraform nvm

```

The screenshot shows the VS Code interface with a Terraform configuration file open. The left sidebar displays the project structure, including files like `index.html`, `main.ts`, `.editorconfig`, `.gitignore`, `.gitlab-ci.yml`, `terraform.lock.hcl`, `angular.json`, `Dev-IMM.json`, `Dockerfile`, `ec2.tf` (which is currently selected), `firebase.json`, `gitignore`, `gueguet-app.pem`, `mini-project-gitlab.pem`, `package.json`, `README.md`, `terraform.tfstate`, and `terraform.tfstate.backup`. The main editor area shows the Terraform code for creating an EC2 instance, and the terminal at the bottom shows the execution of the `terraform apply` command.

```
+ 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-0a69bab54ccf0963]

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

#### • Etape 4 :

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

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed, and the main content area displays the following:

**Instances (1) Informations**

Date de la dernière mise à jour Il y a 5 minutes

Se connecter État de l'instance Actions Lancer des instances

Rechercher Instance par attribut ou identification (case-sensitive)

Tous les états

État de l'instance = running Effacer les filtres

Name	ID d'instance	État de l'instance	Type d'instance	Contrôle des statut	Statut d'alarme	Zone de disponibilité
gueueitapp	i-0a69bab548ccf0963	En cours d'initialisation	t2.medium	Initialisation en cours	Afficher les alarmes	us-east-1b

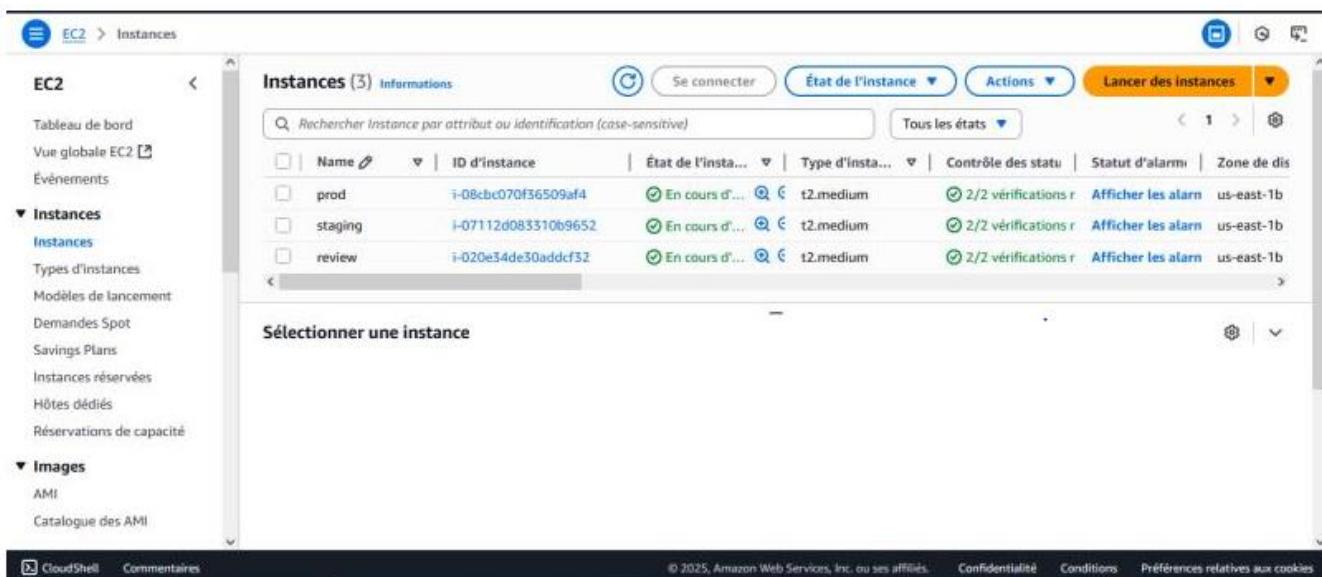
**Sélectionner une instance**

CloudShell Commentaires

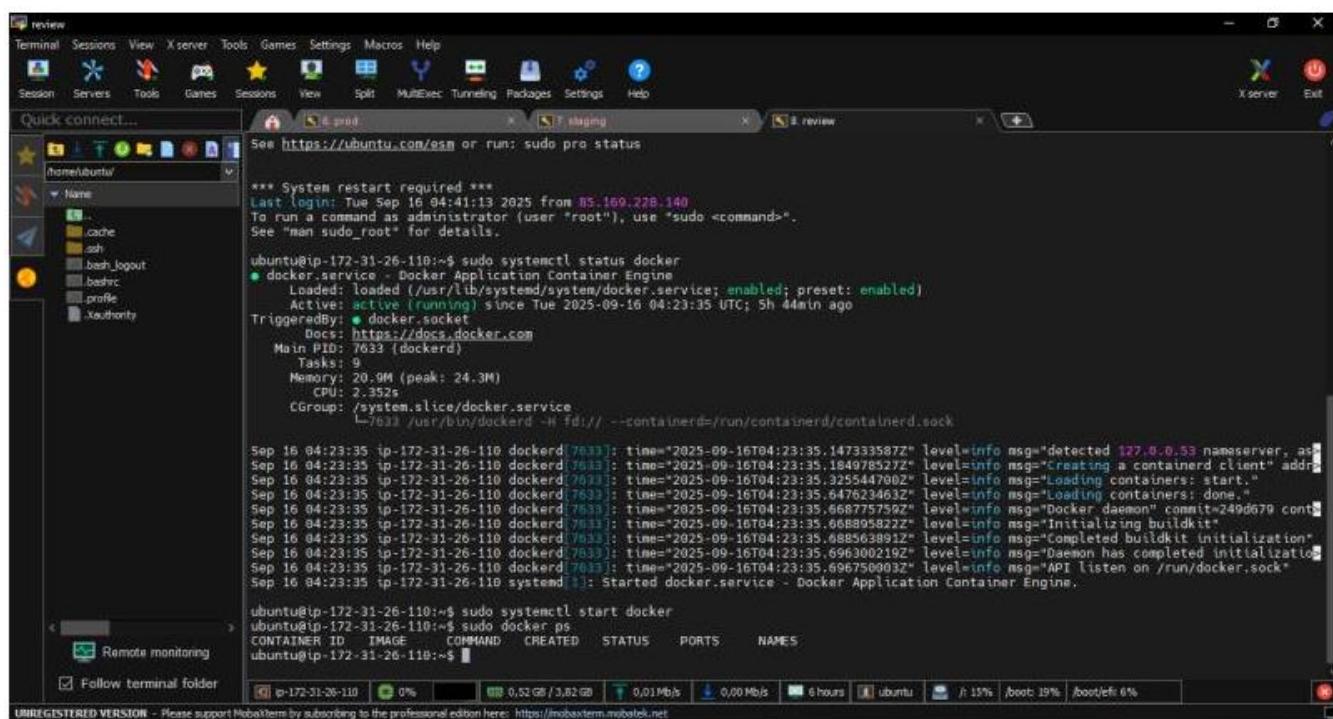
© 2025, Amazon Web Services, Inc. ou ses affiliés. Confidentialité Conditions Préférences relatives aux cookies

## 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».



The screenshot shows the AWS Management Console interface for the EC2 service. On the left, there's a sidebar with navigation links for 'Tableau de bord', 'Vue globale EC2', 'Événements', 'Instances' (selected), 'Types d'instances', 'Modèles de lancement', 'Demandes Spot', 'Savings Plans', 'Instances réservées', 'Hôtes dédiés', 'Réservations de capacité', and 'Images'. The main content area is titled 'Instances (3) Informations' and lists three instances: 'prod' (status: En cours d...', type: t2.medium, 2/2 vérifications réussies), 'staging' (status: En cours d...', type: t2.medium, 2/2 vérifications réussies), and 'review' (status: En cours d...', type: t2.medium, 2/2 vérifications réussies). Below the table, a section titled 'Sélectionner une instance' is visible. At the bottom of the page, there are links for 'CloudShell', 'Commentaires', and several legal and preference links.



The screenshot shows a MobaXterm window with three tabs: 'prod', 'staging', and 'review'. The 'review' tab is active and displays terminal output for setting up Docker. It includes commands like 'sudo apt update', 'sudo apt install docker.io', 'sudo systemctl start docker', and 'sudo systemctl enable docker'. The terminal also shows logs for Docker daemon startup and container status. The MobaXterm interface includes a file manager on the left, a toolbar at the top, and various status indicators at the bottom.

**Project**

- Manage
- Plan
- Automate
- Code
- Build
- Pipelines
- Jobs
- Pipeline editor
- Pipeline schedules
- Artifacts
- Secure
- What's new
- Help

**add ip**

Running Malloum David Guendergue created pipeline for commit 38468bd8 · 4 minutes ago.

For main

latest branch 6 jobs In progress, queued for 1 seconds

**Pipeline** Jobs Tests

```

graph LR
    build[build] --> test[test]
    test --> release[release]
    release --> deploy[deploy]
    
```

**build**: Build

**test**: Test

**release**: Release Image

**deploy**: Deploy Production, Deploy review, Deploy staging

**Project**

- Manage
- Plan
- Automate
- Code
- Build
- Pipelines
- Jobs
- Pipeline editor
- Pipeline schedules
- Artifacts
- Secure
- Deploy
- What's new
- Help

**David Guendergue / Angular-gueguet-appproject / Jobs / #11370411893**

Search visible log output

```

0.0s done
111 #5 extracting sha256:1f55cd1d78f288f4e4b4769f682960cee59c01dc374c66ecc88ba80dba186787
done
112 #5 DONE 22.0s
113 #6 [2/3] WORKDIR /app
114 #6 DONE 3.0s
115 #7 [3/3] COPY index.html .
116 #7 DONE 0.0s
117 #8 exporting to image
118 #8 exporting layers 0.0s done
119 #8 writing image sha256:17cc9a23d85013a39f38e759ad578d8e827ca52195c17ed25ee72554c72ec8
76 done
120 #8 naming to docker.io/library/gueguetapp:v1 done
121 #8 DONE 0.1s
122 $ docker save -o gueguet-app.tar $IMAGE_NAME:$IMAGE_TAG
123 Uploading artifacts for successful job
124 Uploading artifacts...
125 gueguet-app.tar: found 1 matching artifact files and directories
126 Uploading artifacts as "archive" to coordinator... 201 Created correlation_id=d5ce7b7
e34a59dea85db1b159357b20 id=11370411893 responseStatus=201 Created token=6a_EBRT2Q
127 Cleaning up project directory and file based variables
128 Job succeeded

```

Duration: 2 minutes 18 seconds  
 Finished: 13 minutes ago  
 Queued: 0 seconds  
 Timeout: 1h (from project)  
 Runner: #12270857 (nTlFEltyX)-4-green.saaS-linux-small-amd64.runners-manager.gitlab.com/default  
 Source: Push

**Job artifacts** These artifacts are the latest. They will not be deleted (even if expired) until newer artifacts are available.

Keep Download Browse

**Commit** a28b1fab · add ip

**Pipeline** #2042389912 · Failed for main

Instances | EC2 | us-east-1 | TOP 5 des études qui ... | Docker permission de ... | Test (#11370411905) | maloum/david/gueguet | Facebook

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

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

Search or go to... Search visible log output

**Project**

- Manage
- Plan
- Automate
- Code
- Build
- Pipelines
- Jobs**
- Pipeline editor
- Pipeline schedules
- Artifacts
- Secure
- Deploy
- What's new
- Help

```

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 sha256:141d08d5b9ccb9f76108a857ea1357c593dd7ba8d85cae02cbe797b3c59
e69e for docker:latest with digest docker@sha256:831644212c5bdd0b3362b5855c87b98bea39a
83c6evedocf2ced3eced998737a ...
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 (0.14.1-r1)
35 Executing busybox-1.37.0-r18.trigger
36 OK: 41 MiB in 76 packages
37 $ docker load -i gueguetapp:v1
38 Loaded image: gueguetapp:v1
39 $ docker run --rm -dp $HOST_PORT:$CONTAINER_PORT --name $IMAGE_NAME $IMAGE_NAME:$IMAGE_NAME
--tag
40 aafab18fc627d3c8408a294e8a98575f713b9c951a7e6d2a998ba1caf42c61b0
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: a2801fab | 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) | maloum/david/gueguet | Facebook

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

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

Search or go to... Search visible log output

**Project**

- Manage
- Plan
- Automate
- Code
- Build
- Pipelines
- Jobs**
- Pipeline editor
- Pipeline schedules
- Artifacts
- Secure
- Deploy
- What's new
- Help

```

48 1e167435acbc: Preparing
49 df36a4427956: Preparing
50 88466a243658: Preparing
51 ee2af66d91c: Waiting
52 20435ff34d4a: Waiting
53 1e167435acbc: Waiting
54 df36a4427956: Waiting
55 88466a243658: Waiting
56 b94df955776e: Pushed
57 8348ffdfb2114: Pushed
58 1c4b33828f88: Pushed
59 eee2af66d91c: Pushed
60 954ead644fd7: Pushed
61 b6cd13b84fd: Pushed
62 df36a4427956: Pushed
63 88466a243658: Pushed
64 1e167435acbc: Pushed
65 20435ff34d4a: Pushed
66 v1: digest: sha256:f468fb36c0b6fdcb4fe491063b00f772cc83500270b6c62d19b3a5ce2456eaf size: 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: a2801fab | add ip  
 Pipeline #2042389912 Failed for main

release

Related jobs → **Release Image**

```

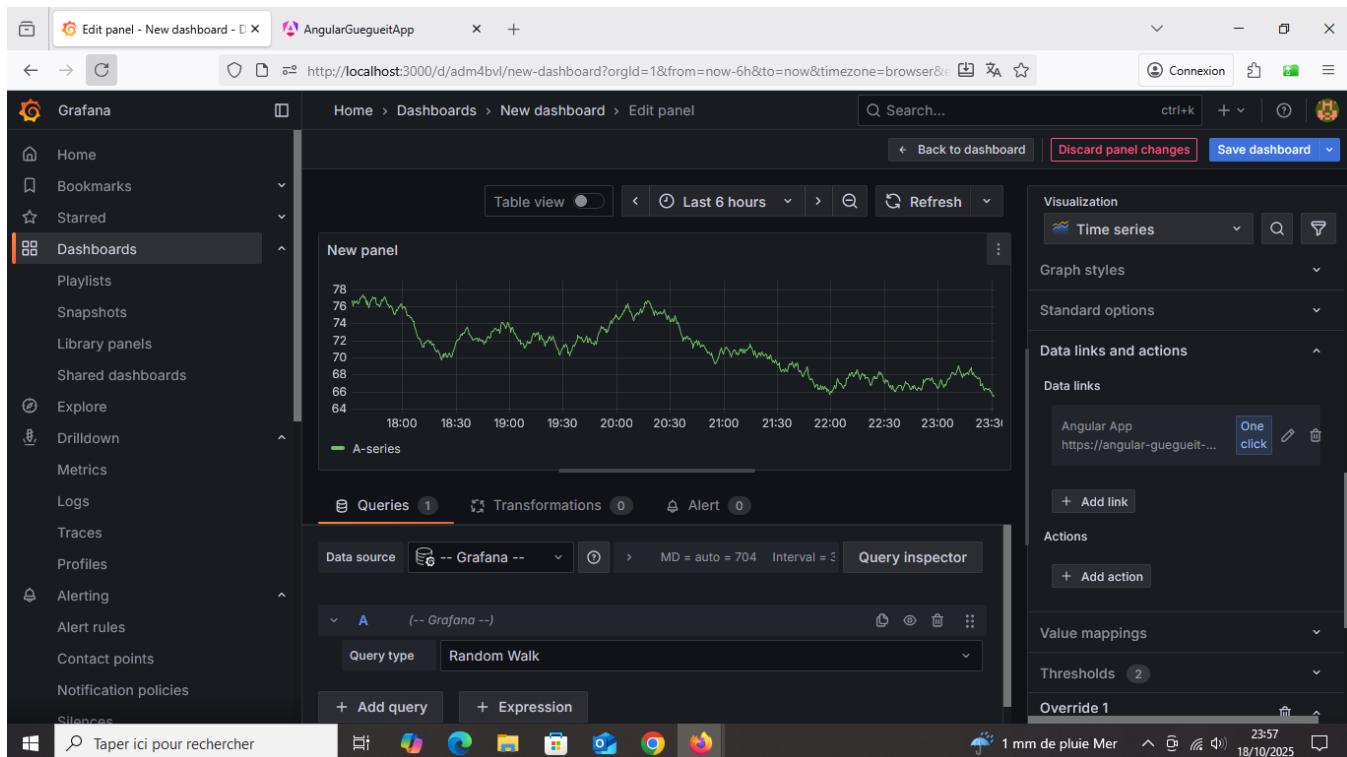
34 (7/10) Installing openssh-sftp-server (10.0_p1-r7)
35 (8/10) Installing openssh-server-common (10.0_p1-r7)
36 (9/10) Installing openssh-server (10.0_p1-r7)
37 (10/10) Installing openssh (10.0_p1-r7)
38 Executing busybox-1.37.0-r18.trigger
39 OK: 14 MiB in 26 packages
40 $ eval $(ssh-agent -s)
41 Agent pid 18
42 $ echo "$SSH_PRIVATE_KEY"
43 SSH_PRIVATE_KEY
44 $ base64 -d > /tmp/id_rsa
45 $ chmod 600 /tmp/id_rsa
46 $ mkdir -p ~/.ssh
47 $ chmod 700 ~/.ssh
48 $ echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
49 $ ssh -o StrictHostKeyChecking=no -i /tmp/id_rsa $SERVER_USERNAME@$SERVER_IP "docker pull $DOCKER_USERNAME/$IMAGE_NAME:$IMAGE_TAG"
50 Warning: Permanently added '52.204.144.30' (ED25519) to the list of known hosts.
51 Load key "/tmp/id_rsa": error in libcrypto
52 ubuntu@52.204.144.30: Permission denied (publickey).
53 Cleaning up project directory and file based variables
54 ERROR: Job failed: exit code 255

```

Duration: 32 seconds  
 Finished: 13 minutes ago  
 Queued: 0 seconds  
 Timeout: 1h (from project)  
 Runner: #32976645 (YKxHNyex) 6-green.saas-linux-small-amd64.runners-manager.gitlab.com/default  
 Source: Push  
 Commit: a2bb1fab  
 add ip  
 Pipeline #2042389912 Failed for main  
 deploy  
 Related jobs  
 → Deploy Production  
 → Deploy review  
 → Deploy staging

## 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 écrit 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<sup>e</sup> 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=CjwKCAjwjffH BhBuEiwAKMb8pEi-pSrJ-Lulw-G1uptkOQngLT\\_87yOrzwwkeO462orfwPll-BxTIBoCO9MQAvD\\_BwE](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=CjwKCAjwjffH BhBuEiwAKMb8pEi-pSrJ-Lulw-G1uptkOQngLT_87yOrzwwkeO462orfwPll-BxTIBoCO9MQAvD_BwE)
- ✚ <https://firebase.google.com/>
- ✚ <https://grafana.com/>
- ✚ <https://calculator.aws/#/>