# Rapport Projet Web

DORCAL Doralie (22208270) & JENY JEYARAJ Jeen (22519719)
M1 Cybersécurité et E-santé

12 janvier 2026

Table des matières

# Introduction

## Structure du dossier

```
DORCAL_JENY_JEYARAJ/
        |----- frontend/
        |         |----- app.py
        |         |----- requirements.txt
        |         |----- Dockerfile
        |         └--- templates
        |                   └--- index.html
        |---- backend/
        |         |----- app.py
        |         |----- requirements.txt
        |         └--- Dockerfile
        |---- kube/
        |         |----- backend-deploy.yml
        |         |----- backend-service.yml
        |         |----- frontend-deploy.yml
        |         |----- frontend-service.yml
        |         |----- ingress.yml
        |         |----- istio-gateway.yml
        |         |----- istio-security.yml
        |         |----- mysql-deployment.yml
        |         |----- mysql-pvc.yml
        |         |----- mysql-service.yml
        |         |----- network-policy.yml
        |         |----- rbac.yml
        |         └--- secrets.yml
        └--- README.md
```

## Présentation du projet

Ce projet consiste à concevoir, déployer et sécuriser une application basée sur une architecture en microservices. L'application est divisée en deux composants principaux développés en Python : un frontend interactif pour l'utilisateur, et un backend chargé de la logique métier. L'ensemble s'appuie sur une base de données locale MySQL pour le stockage et la gestion des données.

Pour garantir un déploiement fiable, flexible et reproductible, chaque composant est conteneurisé à l'aide de Docker, puis orchestré au sein d'un cluster Kubernetes.

L'accès à l'application depuis l'extérieur est géré par une passerelle Ingress Nginx, qui se charge de diriger le trafic vers les bons conteneurs. En parallèle, un composant de type Service Mesh (Istio) est déployé pour contrôler et optimiser la communication interne entre le frontend et le backend.

Cette architecture assure le bon fonctionnement de l'application. Or, nous pouvons aller plus loin en durcissant la sécurité de l'ensemble de l'infrastructure.

Notre application est un recueil de notes; notre petit journal intime. Nous pouvons écrire des notes et les enregistrer. Ces-dernières seront datées et resteront stockées dans la BDD en tant que données persistantes.

## Setup

Nous avons commencé par créer un dossier DORCAL_JENY_JEYARAJ, dans lequel nous allons faire notre projet.

Lien vers le Github du projet : https://github.com/DoralieD/DORCAL_JENY_JEYARAJ
`git clone https://github.com/DoralieD/DORCAL_JENY_JEYARAJ.git`

Dans le terminal, nous nous sommes d'abord connectées au Docker :
`docker login`

Puis, nous avons ouvert la page donnée dans le terminal, afin de saisir le OTP donné et se connecter.

## Lancer l'application

Dans le terminal :

```
minikube start
minikube addons enable ingress
minikube addons enable istio-provisioner
minikube addons enable istio
minikube addons enable ingress-dns
kubectl label namespace default istio-injection=enabled
kubectl apply -f kube/
minikube tunnel
```

Dans votre browser préféré : http://journalintime.info

# I) Création des deux services

## Frontend

Le frontend est comme une vitrine, car il affiche les informations à l'utilisateur et capte ses actions.

Nous avons codé une mini application en python “app.py” et son Dockerfile.

Puis, nous avons créé et tagué une image Docker :
`docker build -t doralie/journal-front:1.0 .`

Ensuite, nous avons publié l’image sur Docker Hub :
`docker push doralie/journal-front:1.0`

Et enfin, nous avons créé un déploiement Kubernetes :
`kubectl apply -f kube/frontend-deploy.yml`

Et enfin, nous avons créé un service Kubernetes :
`kubectl apply -f kube/frontend-service.yml`

## Backend

Le backend reçoit les demandes du frontend, fait les calculs et va chercher les informations.

Nous avons codé une mini application en python “app.py” et son Dockerfile.

Puis, nous avons créé et tagué une image Docker :
`docker build -t doralie/journal-back:1.0 .`

Ensuite, nous avons publié l’image sur Docker Hub :
`docker push doralie/journal-back:1.0`

Et enfin, nous avons créé un déploiement Kubernetes :
`kubectl apply -f kube/backend-deploy.yml`

Et enfin, nous avons créé un service Kubernetes :
`kubectl apply -f kube/backend-service.yml`

# II) Gateway, ingress, nginx, service mesh

## Ingress et nginx

Ceci reçoit les requêtes venant d'Internet et les dirige vers notre frontend.
Ingress : https://github.com/charroux/kubernetes-minikube/blob/main/ingress.yml

Nous avons repris le code et changé le nom en “journal-ingress” et le hostname en “journalintime.info”.

Dans le terminal :
minikube addons enable ingress
kubectl get pods -n ingress-nginx

```
NAME                                        READY   STATUS      RESTARTS      AGE
ingress-nginx-admission-create-6kssn        0/1     Completed   0             5d5h
ingress-nginx-admission-patch-75fcp         0/1     Completed   0             5d5h
ingress-nginx-controller-9cc49f96f-wmmzx    1/1     Running     3 (63s ago)   5d5h
```

kubectl apply -f kube/ingress.yml
kubectl get ingress

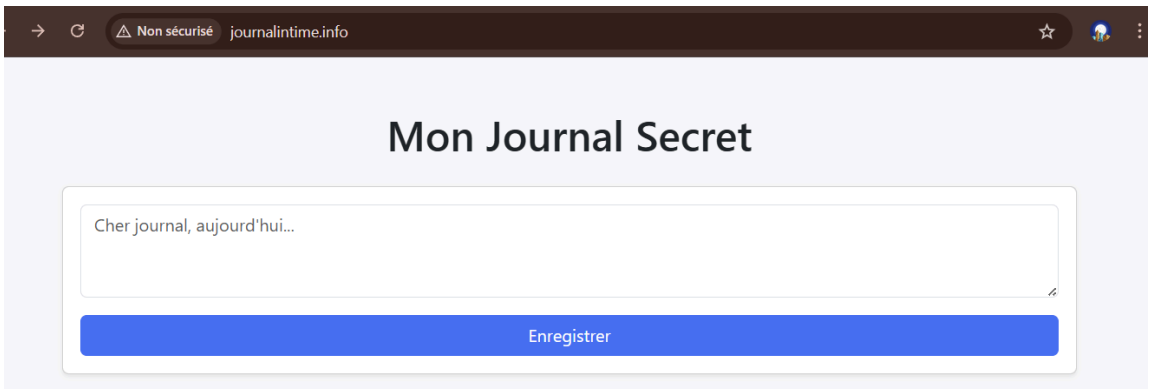
```
NAME              CLASS   HOSTS                ADDRESS        PORTS   AGE
journal-ingress   nginx   journalintime.info   192.168.49.2   80      45m
```

Nous travaillons sur Windows, donc dans c:\windows\system32\drivers\etc\hosts, on ajoute la ligne suivante :
127.0.0.1 journalintime.info

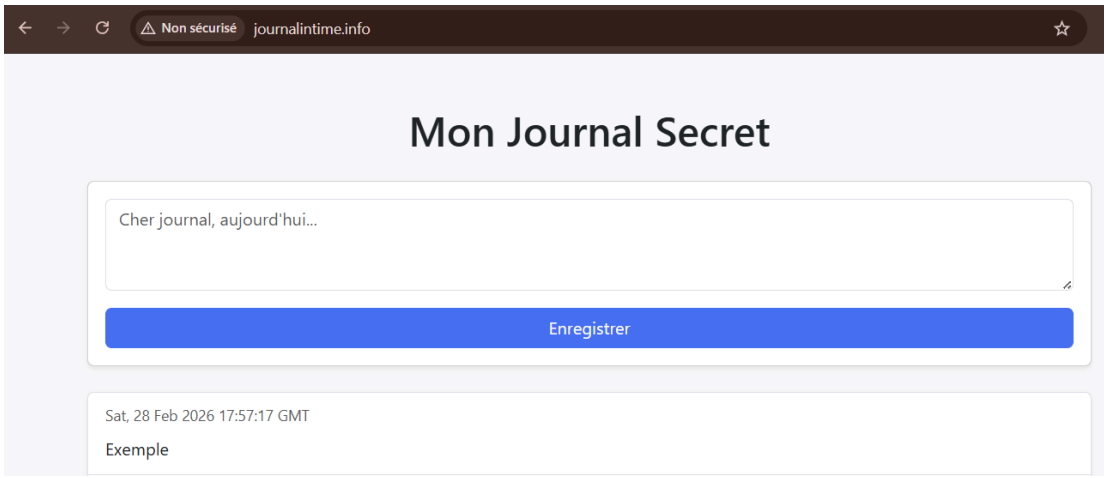
minikube addons enable ingress-dns
minikube tunnel

Puis, nous pouvons accéder à notre site via : http://journalintime.info

Nous voyons que le site fonctionne :



Nous pouvons maintenant nous amuser à ajouter des notes à notre journal intime :




## Istio

Le service mesh gère la communication entre les deux microservices créés et surveille le trafic interne, afin de s'assurer que tout circule bien.

Deux nouveaux fichiers :

- Définir la gateway Istio et les routes vers nos deux services :
  `kubectl apply -f kube/istio-gateway.yml`
- Activer la sécurité Istio (par exemple le mTLS) entre nos services :
- `kubectl apply -f kube/istio-security.yml`

Dans les fichiers de déploiement pour le frontend et le backend, nous avons ajouté une ligne : sidecar.istio.io/inject: "true"

# III) MySQL : une BDD en local

Nous avons aussi besoin d'une base de données locale où le backend va stocker et récupérer les données.

MySQL : https://github.com/charroux/noops/tree/main/mysql
Nous l'avons utilisé et adapté à notre projet.

Lancer le conteneur MySQL dans Kubernetes :
`kubectl apply -f kube/mysql-deployment.yml`

Réserver un stockage persistant pour que MySQL garde les données :
`kubectl apply -f kube/mysql-pvc.yml`

Permet aux autres services du cluster d'accéder à MySQL via un nom stable :
`kubectl apply -f kube/mysql-service.yml`

Contenir les identifiants sensibles comme le mot de passe MySQL :
`kubectl apply -f kube/secrets.yml`

# IV) Sécuriser le cluster

Afin d'assurer la sécurité de l'ensemble de l'infrastructure, nous avons mis en place plusieurs moyens :

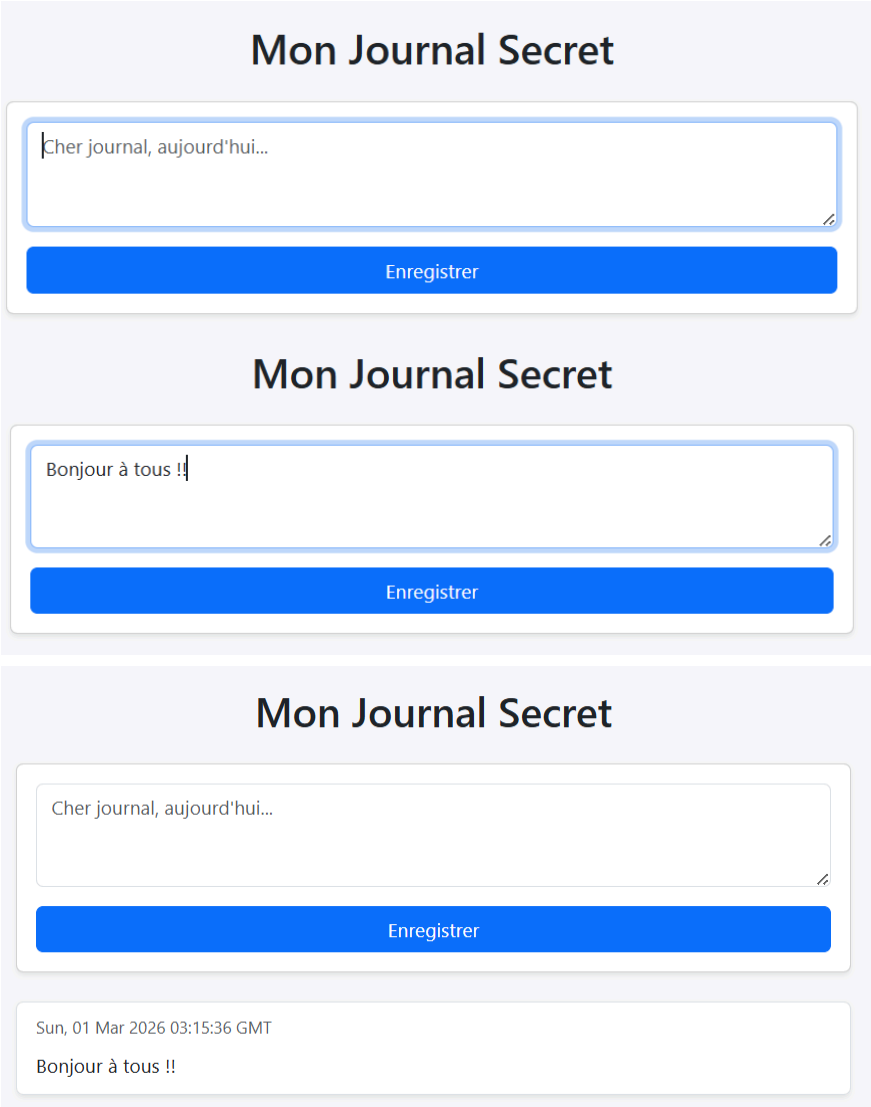
- **Registry image sécurisé (Docker Hub) :**
  En effet, Docker Hub effectue un "Vulnerability Scanning" (scan de failles) sur nos deux images "doralie/journal-back" et "doralie/journal-front" pour s'assurer qu'elles ne contiennent pas de virus ou de librairies périmées.
- **Sécurité Proxy (Service Mesh, Istio) :**
  Dans nos fichiers de déploiement, nous avons ajouté l'annotation qui force l'injection du proxy Envoy, c'est-à-dire le "sidecar", devant nos applications.
- **Cryptage des échanges (Mutual TLS) :**
  Pour que les microservices se parlent de manière cryptée, nous avons un fichier nommé "istio-security.yml" qui force le cryptage mTLS entre nos deux services. Ainsi, personne ne peut interférer entre les messages du frontend et du backend.
- **3 outils de contrôle de la sécurité Kubernetes :**
  - RBAC (Role-based access control) : un mécanisme de contrôle d'accès qui définit les rôles et les autorisations de chaque utilisateur
  - Network Policies : un pare-feu interne qui n'autorise que le backend à parler à la base de données. Ceci empêche qu'un pirate ne vole nos données s'il parvient à s'introduire par le frontend.
  - Secrets : définit le mot de passe qui sera utilisé lors de la connexion au serveur MySQL.

# Conclusion

En conclusion, ce projet nous a permis de suivre l'ensemble des étapes clés du cycle de vie d'une application, de sa conception à sa sécurisation.

Nous avons développé deux microservices en python, le frontend et le backend, en utilisant une base de données MySQL locale. D'une part, Docker et Kubernetes nous ont assuré un déploiement fiable et reproductible. D'autre part, l'intégration d'Ingress Nginx et d'Istio a garanti un accès sécurisé et une communication interne fluide entre les services. De plus, sécuriser le cluster nous a montré l'importance de protéger les données et le fonctionnement de l'application.

Ainsi, ce projet illustre comment combiner efficacité, modularité et sécurité dans une architecture moderne, tout en offrant une application fonctionnelle et interactive : notre propre journal intime numérique.



# PROGWEB

# **Rapports des 2 Labs**

Doralie Dorcal
22208270
Master 1


7 janvier 2026

# Sécuriser les compilations de conteneurs

## Prérequis

Tout d'abord, il a été nécessaire d'activer Cloud Shell. Nous avons ensuite listé les comptes actifs à l'aide de la commande suivante :

- gcloud auth list

```
Use `gcloud config set project [PROJECT_ID]` to change to a different project.
student_04_4cf800f62bb1@cloudshell:~ (qwiklabs-gcp-03-5cf8ff1a2d95)$ gcloud auth list
Credentialed Accounts

ACTIVE: *
ACCOUNT: student-04-4cf800f62bb1@qwiklabs.net

To set the active account, run:
    $ gcloud config set account `ACCOUNT`

student_04_4cf800f62bb1@cloudshell:~ (qwiklabs-gcp-03-5cf8ff1a2d95)$
```

Le compte actif utilisé pour cet atelier est le compte temporaire **student**.

De la même façon, nous avons pu  lister les ID de projet à l'aide de cette commande :

- gcloud config list project

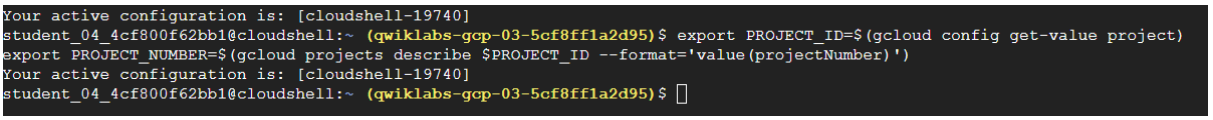
```
student_04_4cf800f62bb1@cloudshell:~ (qwiklabs-gcp-03-5cf8ff1a2d95)$ gcloud config list project
[core]
project = qwiklabs-gcp-03-5cf8ff1a2d95

Your active configuration is: [cloudshell-19740]
student_04_4cf800f62bb1@cloudshell:~ (qwiklabs-gcp-03-5cf8ff1a2d95)$
```

Le projet utilisé dans le cadre de cet atelier est un projet Qwiklabs fourni automatiquement.

# Configuration Workspace

Dans le Cloud Shell, nous avons défini l'ID et le numéro de votre projet en tant que variables externes ( `PROJECT_ID` et `PROJECT_NUMBER` ) avec la commande suivante:

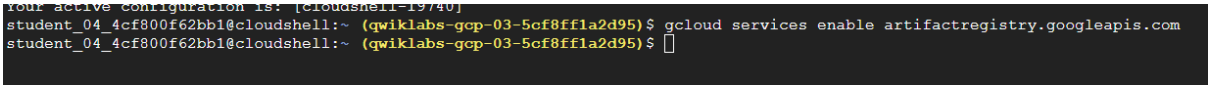
- export PROJECT_ID=$(gcloud config get-value project)
- export PROJECT_NUMBER=$(gcloud projects describe $PROJECT_ID --format='value(projectNumber)')




Ces variables permettent de réutiliser facilement les informations du projet dans les commandes suivantes.

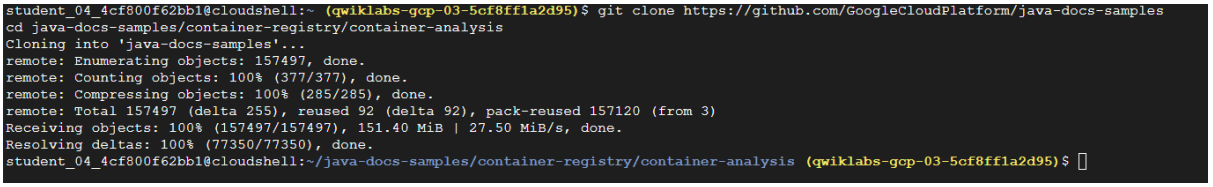
Nous avons ensuite activé l'API **Artifact Registry**, indispensable pour pouvoir créer et utiliser des dépôts d'artefacts :

- gcloud services enable artifactregistry.googleapis.com




Et enfin, nous avons cloné le dépôt nécessaire à l'atelier avec la commande :

- `git clone https://github.com/GoogleCloudPlatform/java-docs-samples`




Après le clonage, nous nous sommes positionnés dans le répertoire suivant : `java-docs-samples/container-registry/container-analysis`

# Dépôts standards

Les dépôts standards dans Artifact Registry permettent de stocker et d'organiser les artefacts de manière centralisée et sécurisée. Dans cet atelier, un dépôt Maven a été créé afin d'héberger des artefacts Java privés.

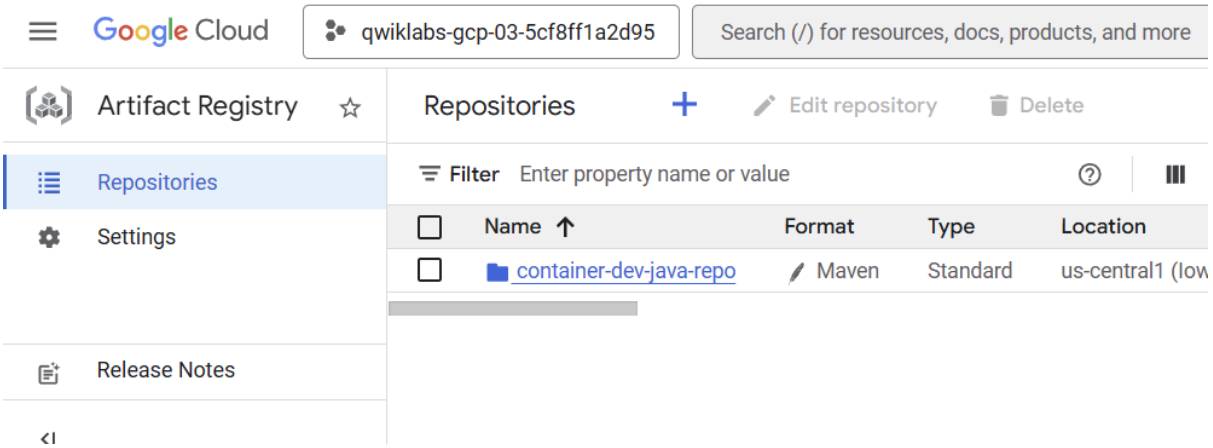
Le dépôt créé porte le nom container-dev-java-repo et est localisé dans la région `us-central1`.

Ce dépôt est visible depuis la console Google Cloud, dans la rubrique Artifact Registry, où il apparaît initialement vide.

Il est également possible d'examiner les informations du dépôt depuis le terminal à l'aide de la commande suivante :

```
student_04_4cf800f62bb1@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-5cf8ff1a2d95)$ gcloud artifacts repositorie
s create container-dev-java-repo \
    --repository-format=maven \
    --location=us-central1 \
    --description="Java package repository for Container Dev Workshop"
Create request issued for: [container-dev-java-repo]
Waiting for operation [projects/qwiklabs-gcp-03-5cf8ff1a2d95/locations/us-central1/operations/97a20d8b-d1e4-4058-8d05-91df9d30d5de] to complete...workin
g..
Waiting for operation [projects/qwiklabs-gcp-03-5cf8ff1a2d95/locations/us-central1/operations/97a20d8b-d1e4-4058-8d05-91df9d30d5de] to complete...done.
Created repository [container-dev-java-repo].
student_04_4cf800f62bb1@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-5cf8ff1a2d95)$
```

Nous pouvons voir que cela s'est réalisé en allant dans la Rubrique Artefact Registry, nous avons créé un dépôt portant le nom container-dev-java-repo



Ensuite, nous avons pu examiner le dépôt depuis le terminal avec la commande qui suit :

- gcloud artifacts repositories describe container-dev-java-repo \
  --location=us-central1

```
student_04_4cf800f62bb1@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-5cf8ff1a2d95)$ gcloud artifacts repositori
es describe container-dev-java-repo \
    --location=us-central1
Encryption: Google-managed key
Repository Size: 0.000MB
createTime: '2026-01-11T16:37:38.400430Z'
description: Java package repository for Container Dev Workshop
format: MAVEN
mode: STANDARD_REPOSITORY
name: projects/qwiklabs-gcp-03-5cf8ff1a2d95/locations/us-central1/repositories/container-dev-java-repo
registryUri: us-central1-maven.pkg.dev/qwiklabs-gcp-03-5cf8ff1a2d95/container-dev-java-repo
satisfiesPzi: true
updateTime: '2026-01-11T16:37:38.400430Z'
vulnerabilityScanningConfig:
  enablementState: SCANNING_DISABLED
  enablementStateReason: Vulnerability scanning is disabled by default for MAVEN repositories.
  lastEnableTime: '2026-01-11T16:37:30.478021602Z'
student_04_4cf800f62bb1@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-5cf8ff1a2d95)$
```

Cette commande permet d'obtenir des informations telles que le type de dépôt, son emplacement, son mode et son état.

# Configurer Maven pour Artifact Registry

Afin de permettre à Maven d'interagir avec Artifact Registry, nous avons généré la configuration du dépôt Maven à l'aide de la commande suivante :

- gcloud artifacts print-settings mvn \
- --repository=container-dev-java-repo \
- --location=us-central1

Ce qui a retourné une configuration en XML destinée à être intégrée dans le fichier `pom.xml` du projet.

```
student_04_4cf800f62bb1@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-5cf8ff1a2d95)$ gcloud artifacts print-set
ings mvn \
    --repository=container-dev-java-repo \
    --location=us-central1
<!-- Insert following snippet into your pom.xml -->

<project>
  <distributionManagement>
    <snapshotRepository>
      <id>artifact-registry</id>
      <url>artifactregistry://us-central1-maven.pkg.dev/qwiklabs-gcp-03-5cf8ff1a2d95/container-dev-java-repo</url>
    </snapshotRepository>
    <repository>
      <id>artifact-registry</id>
      <url>artifactregistry://us-central1-maven.pkg.dev/qwiklabs-gcp-03-5cf8ff1a2d95/container-dev-java-repo</url>
    </repository>
```

Cette configuration comprend :

- la section `<repositories>`, qui indique à Maven où récupérer les dépendances,
- la section `<distributionManagement>`, qui précise le dépôt dans lequel publier les artefacts,
- l'extension `artifactregistry-maven-wagon`, nécessaire pour l'authentification et le transport vers Artifact Registry.

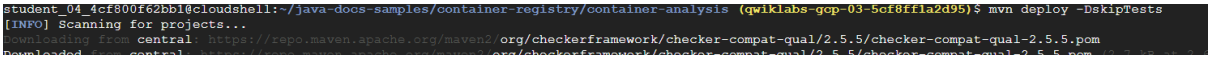
Une fois cette configuration ajoutée au fichier `pom.xml`, Maven est capable de publier et de récupérer des artefacts directement depuis Artifact Registry, ce qui permet de sécuriser et centraliser les dépendances du projet.

La commande suivante a permis d'ouvrir l'éditeur Cloud Shell dans le répertoire courant afin de faciliter l'édition du fichier **pom.xml** :

- cloudshell workspace .

Après l'édition du fichier, les packages ont pu être importés dans Artifact Registry à l'aide de la commande suivante :

- mvn deploy -DskipTests

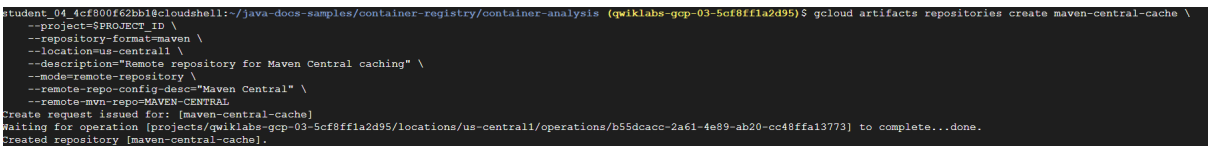



Comme pour la création du dépôt, nous pouvons vérifier directement dans la rubrique Artefact Registry si l'import s'est correctement déroulé.

# Dépôts distants

Un dépôt distant permet de mettre en cache des dépendances provenant de dépôts externes, comme Maven Central.

Pour créer un dépôt distant, on s'y prend avec la commande suivante :



Ensuite, nous avons pu examiner le dépôt depuis le terminal avec la commande suivante :

- gcloud artifacts repositories describe maven-central-cache \
--location=us-central1

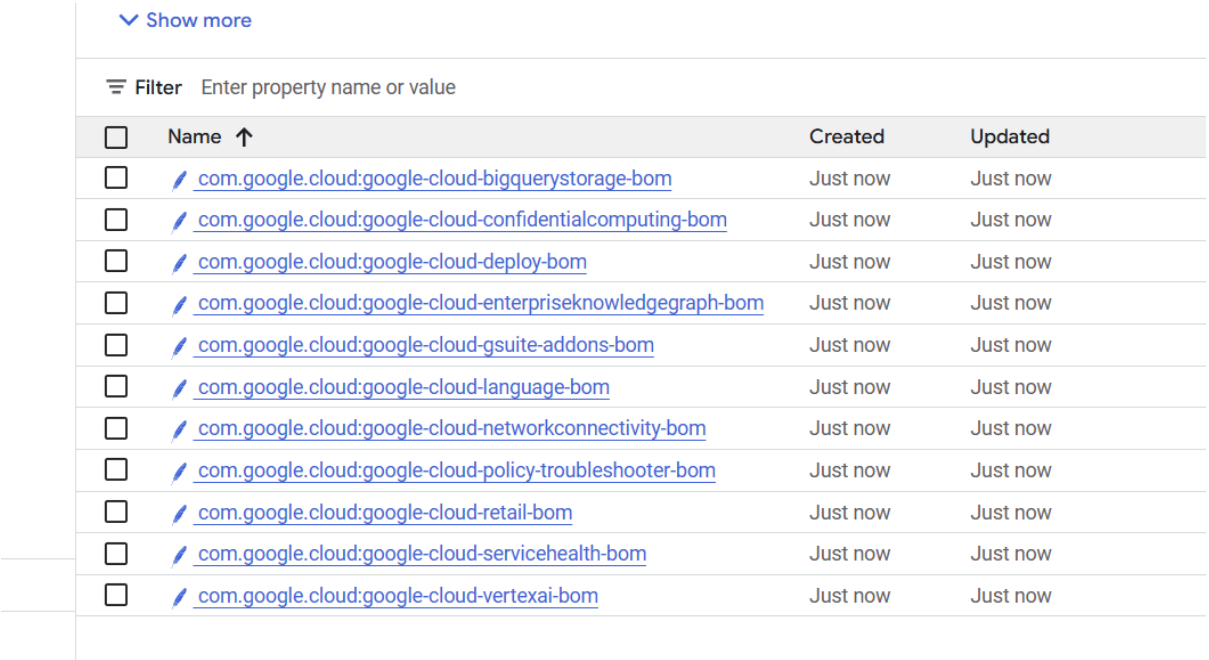
Cette commande retourne une configuration XML à intégrer dans le fichier **pom.xml**. Dans ce cas, seule la section correspondant au dépôt distant a été ajoutée.

Il a également été nécessaire de créer un fichier **extensions.xml** pour le projet. Ce fichier permet d'utiliser le mécanisme d'extensions Maven et garantit que les dépendances parentes ou de plugins peuvent être résolues depuis Artifact Registry

Une fois cette configuration effectuée, nous avons compilé l'application à l'aide des commandes suivantes :

➔ rm -rf ~/.m2/repository
➔ mvn compile

Pour vérifier que cela s'est bien réalisé, dans la console Cloud, il faut accéder à Artifact Registry > Dépôts. En cliquant sur maven-central-cache, nous pouvons constater leur présence.



# Tâche 4 : Dépôts virtuels

Les dépôts virtuels permettent d'accéder à plusieurs dépôts avec une seule configuration. Cela facilite la configuration côté client pour utiliser nos artefacts.

Nous avons d'abord créé un fichier de règles policy.json

```
[
  {
    "id": "private",
    "repository": "projects/${PROJECT_ID}/locations/us-central1/repositories/container-dev-java-repo",
    "priority": 100
  },
  {
    "id": "central",
    "repository": "projects/${PROJECT_ID}/locations/us-central1/repositories/maven-central-cache",
    "priority": 80
  }
]
```

Puis, nous avons créé le dépôt virtuel  avec la commande :

- ➔ gcloud artifacts repositories create virtual-maven-repo \
- ➔ --project=${PROJECT_ID} \
- ➔ --repository-format=maven \
- ➔ --mode=virtual-repository \
- ➔ --location=us-central1 \
- ➔ --description="Virtual Maven Repo" \
- ➔ --upstream-policy-file=./policy.json

Nous avons ensuite  exécuté la commande suivante pour afficher la configuration de dépôt à ajouter à notre projet Java :

- ➔ gcloud artifacts print-settings mvn \
- ➔ --repository=virtual-maven-repo \
- ➔ --location=us-central1

Nous avons ensuite remplacé l'intégralité de la section "repositories" du fichier pom.xml par la section fournie par la sortie de la commande.

# Extraire les dépendances du dépôt virtuel

Étant donné que le dépôt virtuel est un dépôt intermédiaire et ne stocke aucun package réel, pour illustrer clairement le processus, nous avons supprimé le dépôt maven-central-cache créé précédemment avant de le recréer, afin de repartir d'un dépôt vide.
Les commandes suivantes ont été exécutées pour recréer le dépôt du cache :

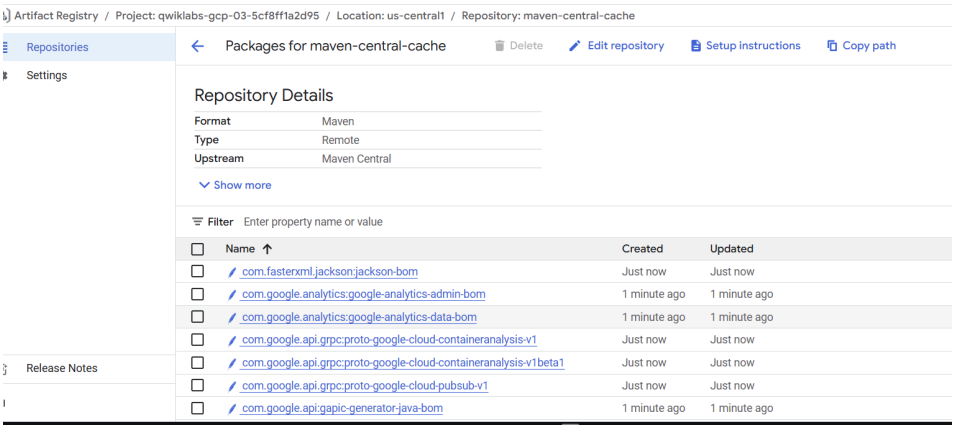
```
➔ gcloud artifacts repositories delete maven-central-cache \
➔ --project=$PROJECT_ID \
➔ --location=us-central1 \
➔ --quiet
➔
➔ gcloud artifacts repositories create maven-central-cache \
➔ --project=$PROJECT_ID \
➔ --repository-format=maven \
➔ --location=us-central1 \
➔ --description="Remote repository for Maven Central caching" \
➔ --mode=remote-repository \
➔ --remote-repo-config-desc="Maven Central" \
➔ --remote-mvn-repo=MAVEN-CENTRAL
```

Il est possible d'examiner le dépôt vide dans la console : Artifact Registry > Dépôts.

Nous avons ensuite testé le dépôt virtuel en compilant le projet à l'aide des commandes suivantes :

```
➔ rm -rf ~/.m2/repository
➔ mvn compile
```

Enfin, nous avons examiné les packages dans la console : Artifact Registry > Dépôts. En cliquant sur `maven-central-cache`, nous avons vérifié que les artefacts binaires avaient été configurés pour être extraits du dépôt virtuel, et qu'ils ont finalement bien été extraits de `maven-central-cache`.



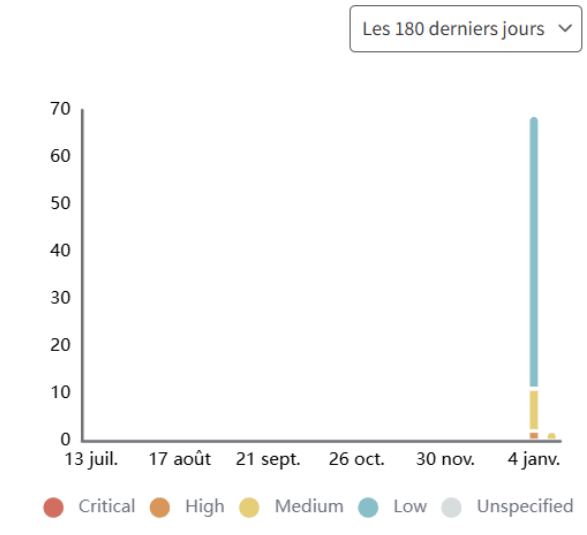
# Get Started with Security Command Center

## Explorer les éléments de l'interface de SCC

Nous avons commencé par accéder au tableau de bord Aperçu des risques via le menu Sécurité de la console Google Cloud. Cette interface nous a permis de visualiser deux types de résultats distincts :

- Menaces : Activités suspectes en cours (ex: compte de service vérifiant ses propres autorisations).
- Failles (Vulnerabilités) : Erreurs de configuration ou logiciels obsolètes (ex: ports ouverts).

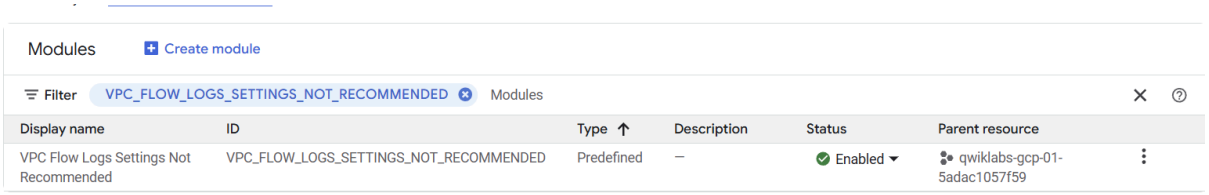
Nous avons observé la répartition des failles par niveau de gravité (Critique, Élevé, Moyen, Faible) et noté la présence de failles de gravité Élevée liées aux ports RDP et SSH ouverts sur le réseau par défaut.

# Configurer les paramètres SCC au niveau du projet

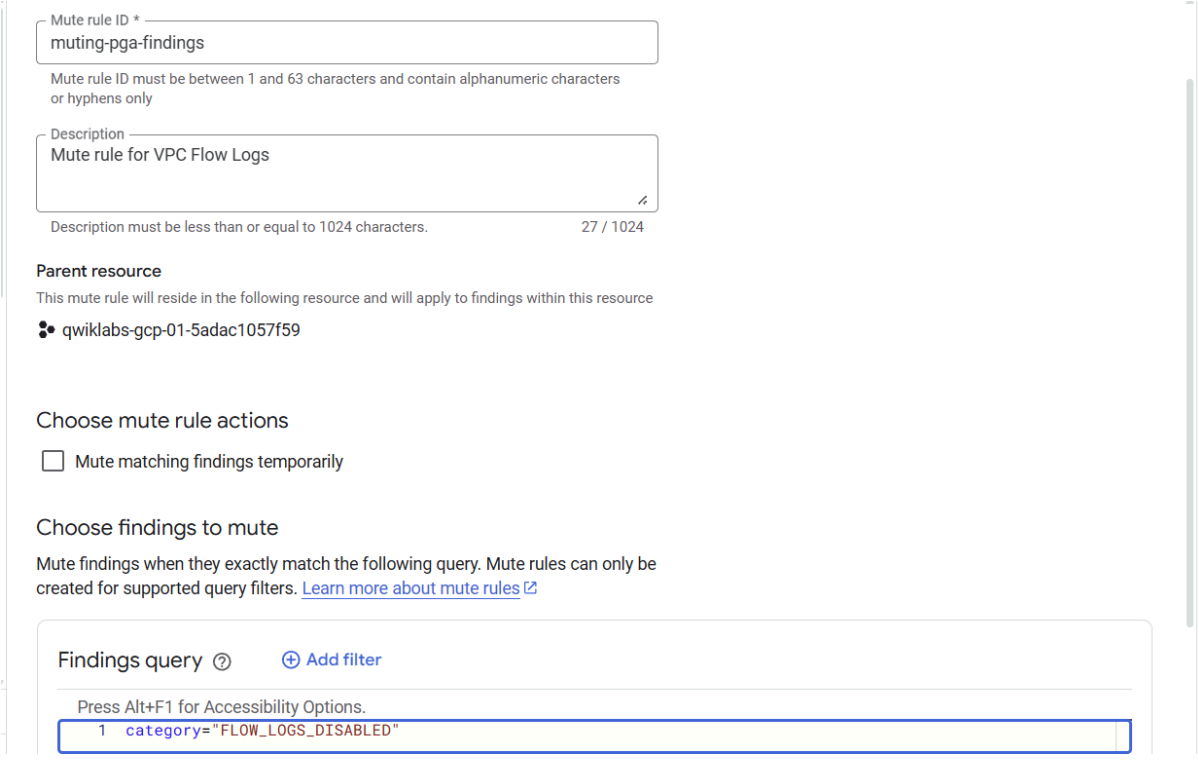
Afin d'affiner la détection des problèmes de sécurité, nous avons configuré le service Security Health Analytics.

Dans les paramètres, sous l'onglet "Modules", nous avons recherché et activé manuellement le module VPC_FLOW_LOGS_SETTINGS_NOT_RECOMMENDED. Ce module permet à SCC de vérifier si les journaux de flux VPC (VPC Flow Logs) sont activés pour les sous-réseaux



# Analyser et corriger les failles identifiées par SCC

Le réseau par défaut générait de nombreuses alertes concernant l'accès privé à Google et les journaux de flux désactivés. Comme il s'agit d'un environnement de test, nous avons décidé d'ignorer ces alertes pour réduire le "bruit".
Nous avons créé une règle d'exclusion nommée muting-pga-findings pour masquer automatiquement les résultats de catégorie FLOW_LOGS_DISABLED.




Pour vérifier le bon fonctionnement de cette règle, nous avons créé un nouveau réseau VPC via Cloud Shell avec la commande :

➔ `gcloud compute networks create scc-lab-net --subnet-mode=auto`

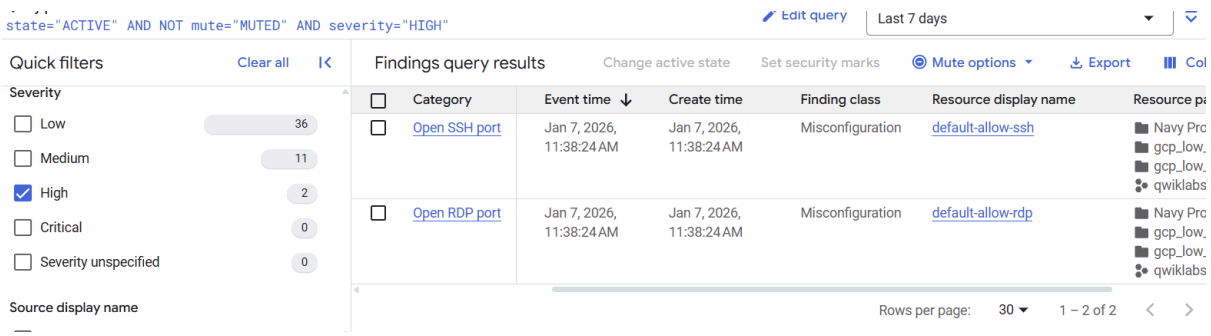
Nous avons constaté que les alertes concernant ce nouveau réseau étaient bien masquées automatiquement par notre règle.

```
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
student_04_4cf800f62bb1@cloudshell:~ (qwiklabs-gcp-01-5adac1057f59)$ gcloud compute networks create scc-lab-net --subnet-mode=auto
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-5adac1057f59/global/networks/scc-lab-net].
NAME: scc-lab-net
SUBNET_MODE: AUTO
BGP_ROUTING_MODE: REGIONAL
IPV4_RANGE:
GATEWAY_IPV4:
INTERNAL_IPV6_RANGE:

Instances on this network will not be reachable until firewall rules
are created. As an example, you can allow all internal traffic between
instances as well as SSH, RDP, and ICMP by running:

$ gcloud compute firewall-rules create <FIREWALL_NAME> --network scc-lab-net --allow tcp,udp,icmp --source-ranges <IP_RANGE>
$ gcloud compute firewall-rules create <FIREWALL_NAME> --network scc-lab-net --allow tcp:22,tcp:3389,icmp

student_04_4cf800f62bb1@cloudshell:~ (qwiklabs-gcp-01-5adac1057f59)$
```

# Remédiation des failles critiques

Comme nous pouvons le voir dans les findings, SCC a identifié deux failles de gravité Élevée sur le pare-feu du réseau par défaut :

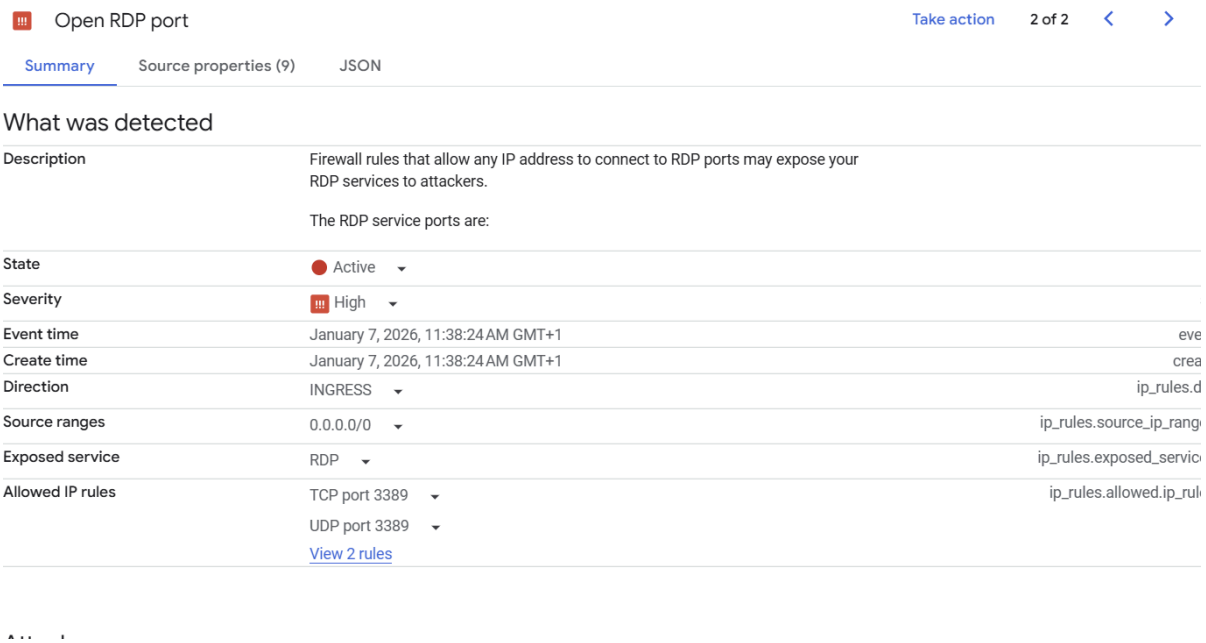
1. Port RDP ouvert
2. Port SSH ouvert



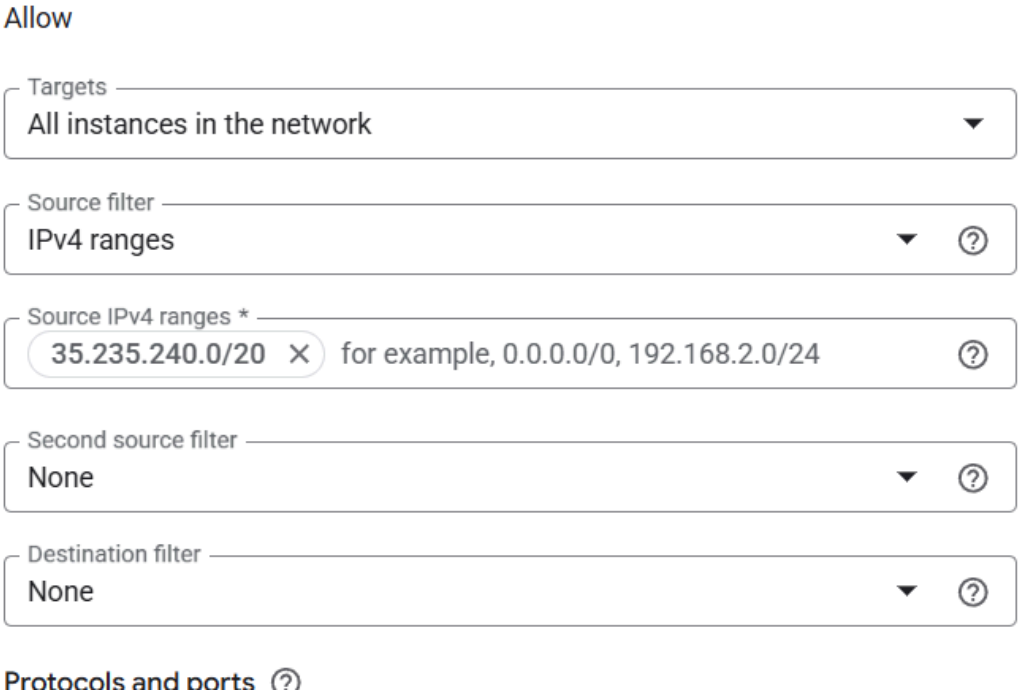

Le problème venait du fait que la plage d'adresses IP source autorisée était 0.0.0.0/0, exposant ainsi les machines à tout l'internet.

Résolvons le problème de port de RDP:

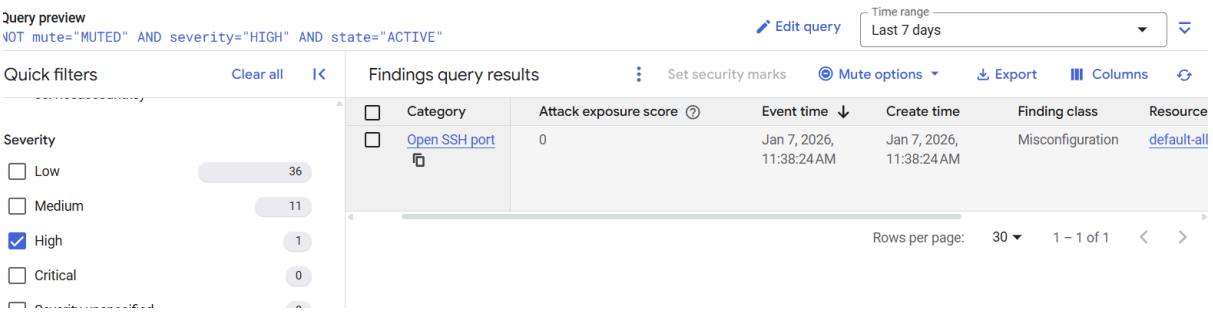
En cliquant sur la Query RDP, cela nous ouvre une fenetre decrivant en details le problème rencontré.




Pour corriger cela, nous avons modifié les règles de pare-feu correspondantes en remplaçant 0.0.0.0/0 par la plage IP 35.235.240.0/20 qui permet une connexion sécurisée et authentifiée aux VM sans les exposer directement.




Une fois la modification enregistrée, nous avons actualisé le tableau de bord SCC et vérifié que la faille de gravité élevée pour le port RDP avait disparu.




Il a fallu répéter la même procédure pour SSH pour la faire disparaître à son tour

# LABS GOOGLE

| Activity | Type | Date started | Date finished | Score | Passed |
|---|---|---|---|---|---|
| How to Use a Network Policy on Google Kubernetes Engine | Lab | 18 minutes ago | 0 minutes ago | Assessment: 100% | ✓ |
| Get Started with Security Command Center | Lab | Jan 11, 2026 | Jan 11, 2026 | Assessment: 100% | ✓ |
| Securing Container Builds | Lab | Jan 11, 2026 | Jan 11, 2026 | Assessment: 100% | ✓ |
| Infrastructure as Code with Terraform | Lab | Dec 15, 2025 | Dec 15, 2025 | Assessment: 100% | ✓ |
| Infrastructure as Code with Terraform | Lab | Dec 15, 2025 | Dec 15, 2025 | Assessment: 70% | |
| Build Infrastructure with Terraform on Google Cloud | Course | Dec 15, 2025 | | | |
| A Tour of Google Cloud Hands-on Labs | Lab | Dec 15, 2025 | Dec 15, 2025 | Assessment: 100% | ✓ |

# Securing Container Builds

JENY JEYARAJ Jeen (22519719)
M1 Cybersécurité et E-santé

12 janvier 2026

Table des matières

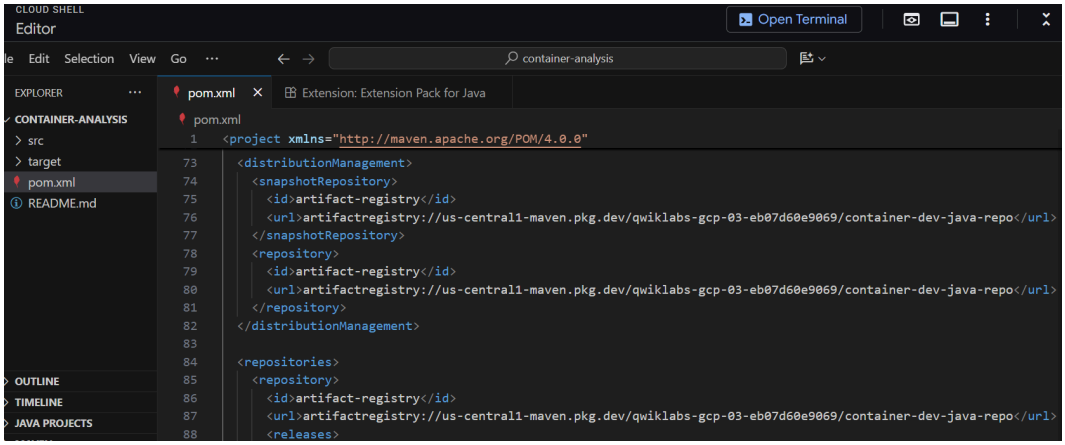
# I) Répertoires standard

Les référentiels standard stockent nos paquets privés et les partagent avec nos autres applications.

```
student_03_d9c3d57dd146@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-eb07d6
0e9069)$ gcloud artifacts repositories create container-dev-java-repo \
    --repository-format=maven \
    --location=us-central1 \
    --description="Java package repository for Container Dev Workshop"
Create request issued for: [container-dev-java-repo]
Waiting for operation [projects/qwiklabs-gcp-03-eb07d60e9069/locations/us-central1/operations/8cce1c47-f452-4a8d-93
47-cd804f9cd504] to complete...done.
Created repository [container-dev-java-repo].
```

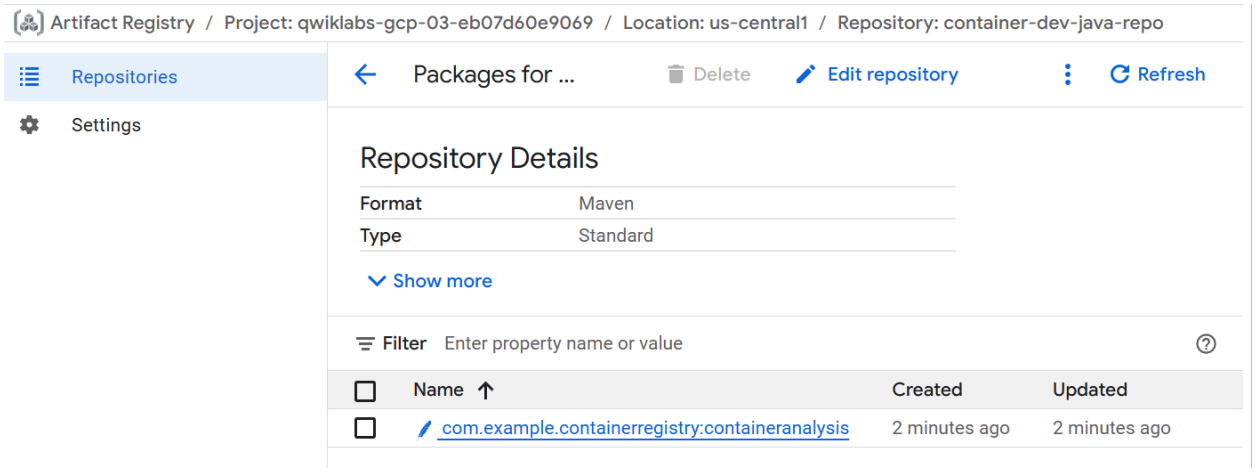
```
student_03_d9c3d57dd146@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-eb07d6
0e9069)$ gcloud artifacts repositories describe container-dev-java-repo \
    --location=us-central1
Encryption: Google-managed key
Repository Size: 0.000MB
createTime: '2026-01-12T05:21:38.032130Z'
description: Java package repository for Container Dev Workshop
format: MAVEN
mode: STANDARD_REPOSITORY
name: projects/qwiklabs-gcp-03-eb07d60e9069/locations/us-central1/repositories/container-dev-java-repo
registryUri: us-central1-maven.pkg.dev/qwiklabs-gcp-03-eb07d60e9069/container-dev-java-repo
satisfiesPzi: true
updateTime: '2026-01-12T05:21:38.032130Z'
vulnerabilityScanningConfig:
  enablementState: SCANNING_DISABLED
  enablementStateReason: Vulnerability scanning is disabled by default for MAVEN repositories.
  lastEnableTime: '2026-01-12T05:21:29.879778452Z'
```

# II) Configurer Maven pour Artifact Registry




```
student_03_d9c3d57dd146@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-eb07d6
0e9069)$ mvn deploy -DskipTests
```

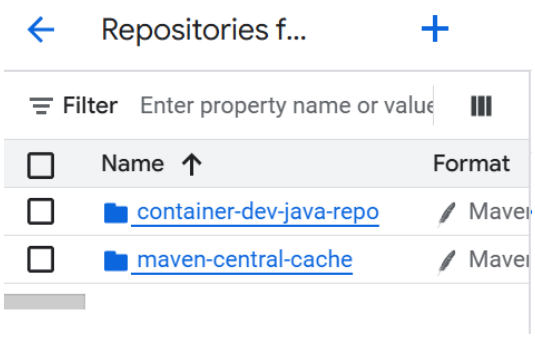
```
dev-java-repo/com/example/containerregistry/containeranalysis/1.0/containeranalysis-1.0.jar
Uploaded to artifact-registry: artifactregistry://us-central1-maven.pkg.dev/qwiklabs-gcp-03-eb07d60e9069/container-d
ev-java-repo/com/example/containerregistry/containeranalysis/1.0/containeranalysis-1.0.jar (19 kB at 18 kB/s)
Downloading from artifact-registry: artifactregistry://us-central1-maven.pkg.dev/qwiklabs-gcp-03-eb07d60e9069/contai
ner-dev-java-repo/com/example/containerregistry/containeranalysis/maven-metadata.xml
Uploading to artifact-registry: artifactregistry://us-central1-maven.pkg.dev/qwiklabs-gcp-03-eb07d60e9069/container-
dev-java-repo/com/example/containerregistry/containeranalysis/maven-metadata.xml
Uploaded to artifact-registry: artifactregistry://us-central1-maven.pkg.dev/qwiklabs-gcp-03-eb07d60e9069/container-d
ev-java-repo/com/example/containerregistry/containeranalysis/maven-metadata.xml (322 B at 317 B/s)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:50 min
```




# III) Répertoires distants

Les répertoires distants permettent de mettre en cache des paquets tiers afin d'améliorer la fiabilité et la sécurité.

```
student_03_d9c3d57dd146@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-eb07d6
0e9069)$ gcloud artifacts repositories create maven-central-cache \
    --project=$PROJECT_ID \
    --repository-format=maven \
    --location=us-central1 \
    --description="Remote repository for Maven Central caching" \
    --mode=remote-repository \
    --remote-repo-config-desc="Maven Central" \
    --remote-mvn-repo=MAVEN-CENTRAL
Create request issued for: [maven-central-cache]
Waiting for operation [projects/qwiklabs-gcp-03-eb07d60e9069/locations/us-central1/operations/d6ccf017-2e53-48d3-b8
f2-0685f930592a] to complete...done.
Created repository [maven-central-cache].
```




```
student_03_d9c3d57dd146@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-eb07d6
0e9069)$ gcloud artifacts repositories describe maven-central-cache \
    --location=us-central1
Encryption: Google-managed key
Repository Size: 0.000MB
createTime: '2026-01-12T05:48:46.390716Z'
description: Remote repository for Maven Central caching
format: MAVEN
mode: REMOTE_REPOSITORY
name: projects/qwiklabs-gcp-03-eb07d60e9069/locations/us-central1/repositories/maven-central-cache
registryUri: us-central1-maven.pkg.dev/qwiklabs-gcp-03-eb07d60e9069/maven-central-cache
remoteRepositoryConfig:
  description: Maven Central
  mavenRepository:
    publicRepository: MAVEN_CENTRAL
satisfiesPzi: true
updateTime: '2026-01-12T05:48:46.390716Z'
vulnerabilityScanningConfig:
  enablementState: SCANNING_DISABLED
  enablementStateReason: Vulnerability scanning is disabled by default for MAVEN repositories.
  lastEnableTime: '2026-01-12T05:48:45.788963922Z'
```

Répertoire ajouté et renommé “central” :

```
<repository>
  <id>central</id>
  <url>artifactregistry://us-central1-maven.pkg.dev/qwiklabs-gcp-03-eb07
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>true</enabled>
  </snapshots>
</repository>
```

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/ow2/asm/asm/9.4/asm-9.4.jar (122 kB at 2.7 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.13.0/plexus
s)
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.13.0/ple
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.13.0/pl
t 88 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox/2.0.3/qdox-2.0.3.jar (334
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.5.0/plexus-utils-3
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.13.0/plex
 kB/s)
[INFO] Changes detected - recompiling the module! :dependency
[INFO] Compiling 13 source files with javac [debug target 1.8] to target/classes
[WARNING] bootstrap class path not set in conjunction with -source 8
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  04:27 min
```

Repository Details

| | |
|---|---|
| Format | Maven |
| Type | Remote |
| Upstream | Maven Central |

Show more

Filter Enter property name or value

| Name | Created | Updated |
|---|---|---|
| com.github.spotbugs:spotbugs-annotations | 1 minute ago | 1 minute ago |
| com.google.android:annotations | 2 minutes ago | 2 minutes ago |
| com.google.api-client:google-api-client-bom | 1 minute ago | 1 minute ago |
| com.google.api.grpc:proto-google-cloud-containeranalysis-v1 | 3 minutes ago | 3 minutes ago |
| com.google.api.grpc:proto-google-cloud-containeranalysis-v1beta1 | 2 minutes ago | 2 minutes ago |
| com.google.api.grpc:proto-google-cloud-pubsub-v1 | 1 minute ago | 1 minute ago |
| com.google.api.grpc:proto-google-common-protos | 3 minutes ago | 3 minutes ago |
| com.google.api.grpc:proto-google-iam-v1 | 1 minute ago | 1 minute ago |
| com.google.api:api-common | 3 minutes ago | 3 minutes ago |
| com.google.api:gax | 2 minutes ago | 2 minutes ago |

# IV) Répertoires virtuels

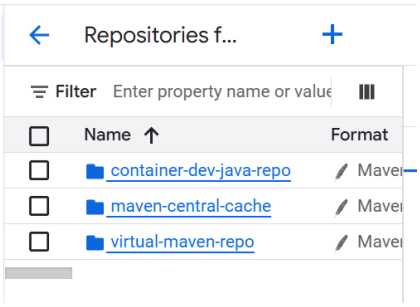
Les référentiels virtuels servent d'interface permettant d'accéder à plusieurs référentiels à partir d'une seule configuration. Cela simplifie la configuration client pour les utilisateurs de nos artefacts et renforce la sécurité en atténuant les attaques par confusion de dépendances.

```
student_03_d9c3d57dd146@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-eb07d6
0e9069)$ gcloud artifacts repositories create virtual-maven-repo \
    --project=${PROJECT_ID} \
    --repository-format=maven \
    --mode=virtual-repository \
    --location=us-central1 \
    --description="Virtual Maven Repo" \
    --upstream-policy-file=./policy.json
Create request issued for: [virtual-maven-repo]
Waiting for operation [projects/qwiklabs-gcp-03-eb07d60e9069/locations/us-central1/operations/4ecdc5ca-f352-4cf0-863
e-f60f4cccd842] to complete...done.
Created repository [virtual-maven-repo].
```

Les répertoires sont remplacés par un répertoire virtuel :

```
84    <repositories>
85      <repository>
86        <id>artifact-registry</id>
87        <url>artifactregistry://us-central1-maven.pkg.dev/qwiklabs-gcp-03-eb07
88        <releases>
89          <enabled>true</enabled>
90        </releases>
91        <snapshots>
92          <enabled>true</enabled>
93        </snapshots>
94      </repository>
95    </repositories>
```

```
student_03_d9c3d57dd146@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-eb07d6
0e9069)$ gcloud artifacts repositories delete maven-central-cache \
    --project=$PROJECT_ID \
    --location=us-central1 \
    --quiet
Delete request issued for: [maven-central-cache]
Waiting for operation [projects/qwiklabs-gcp-03-eb07d60e9069/locations/us-central1/operations/2059172d-2e70-4e61-b3
aa-52176dfe19f6] to complete...done.
Deleted repository [maven-central-cache].
student_03_d9c3d57dd146@cloudshell:~/java-docs-samples/container-registry/container-analysis (qwiklabs-gcp-03-eb07d6
0e9069)$ gcloud artifacts repositories create maven-central-cache \
    --project=$PROJECT_ID \
    --repository-format=maven \
    --location=us-central1 \
    --description="Remote repository for Maven Central caching" \
    --mode=remote-repository \
    --remote-repo-config-desc="Maven Central" \
    --remote-mvn-repo=MAVEN-CENTRAL
Create request issued for: [maven-central-cache]
Waiting for operation [projects/qwiklabs-gcp-03-eb07d60e9069/locations/us-central1/operations/d2001d0e-ddfd-4868-95
2d-02d7c8f94e32] to complete...done.
Created repository [maven-central-cache].
```




```
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.13
us-compiler-javac-2.13.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.13.0/
compiler-api-2.13.0.jar (27 kB at 1.0 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.1
xus-compiler-manager-2.13.0.jar (4.7 kB at 99 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox/2.0.3/qdox-2.0.3.jar
B at 6.2 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.5.0/plexus-u
5.0.jar (267 kB at 5.5 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.13.
s-compiler-javac-2.13.0.jar (23 kB at 456 kB/s)
[INFO] Changes detected - recompiling the module! :dependency
[INFO] Compiling 13 source files with javac [debug target 1.8] to target/classes
[WARNING] bootstrap class path not set in conjunction with -source 8
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  07:34 min
```

# Conclusion

Ce Google Lab m'a permis d'apprendre 3 compétences :

- Utiliser les répertoires standard pour déployer des paquets privés
- Utiliser les répertoires distants pour mettre en cache les paquets Maven Central
- Utiliser les répertoires virtuels pour combiner plusieurs référentiels en amont dans une seule configuration

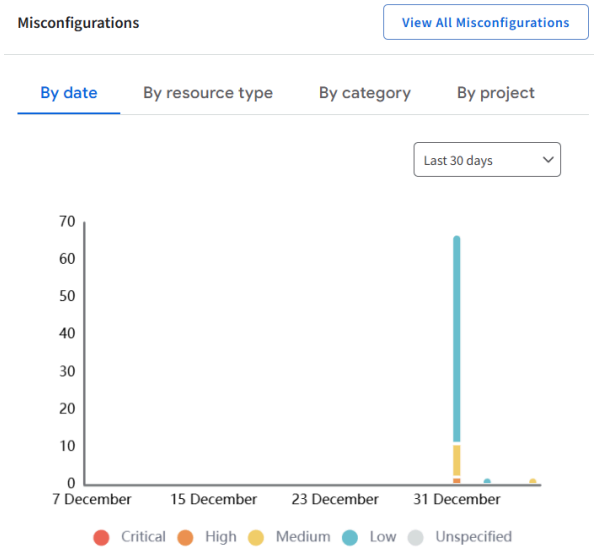
# Security Command Center

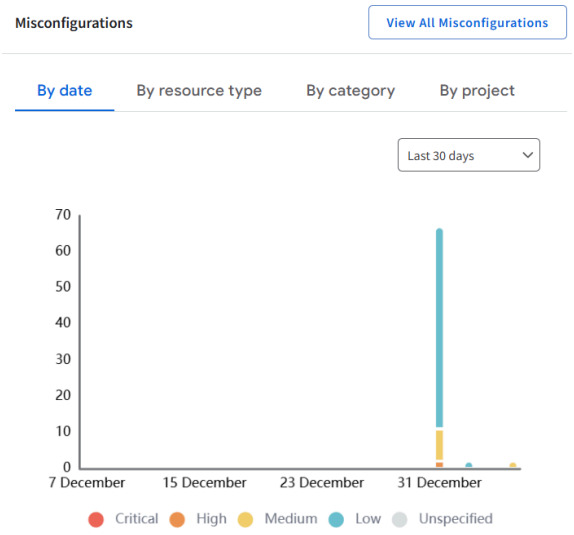
JENY JEYARAJ Jeen (22519719)
M1 Cybersécurité et E-santé

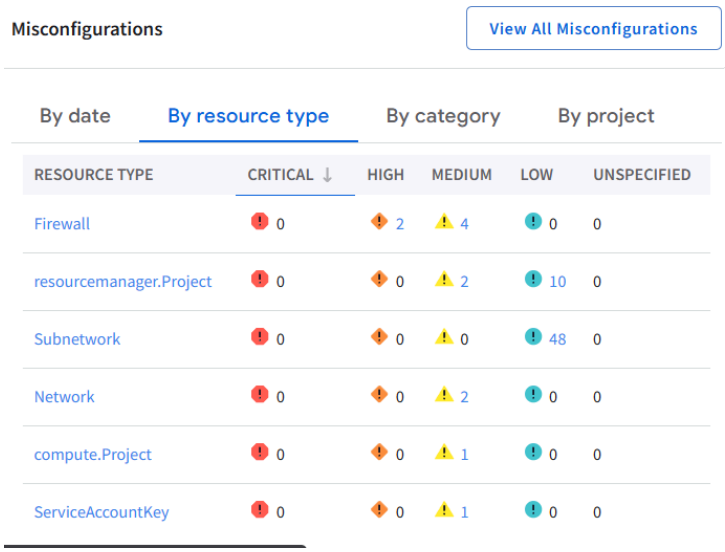
12 janvier 2026

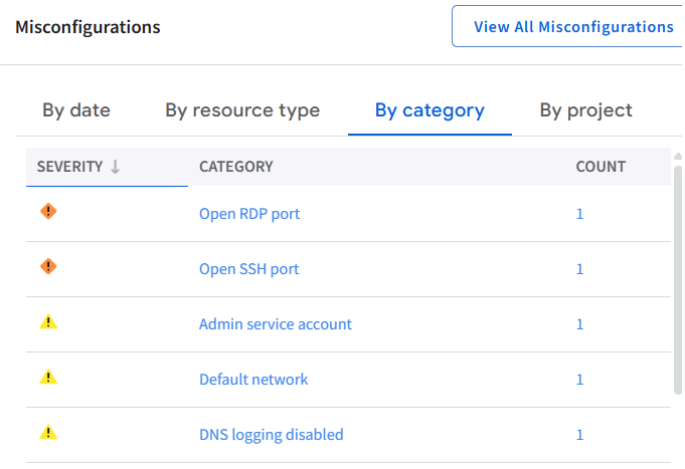
Table des matières

# I) Explorer les éléments de l'interface SCC

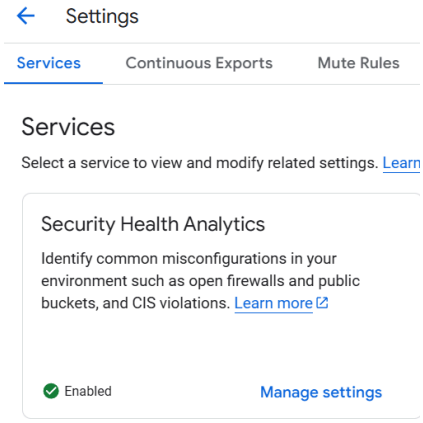


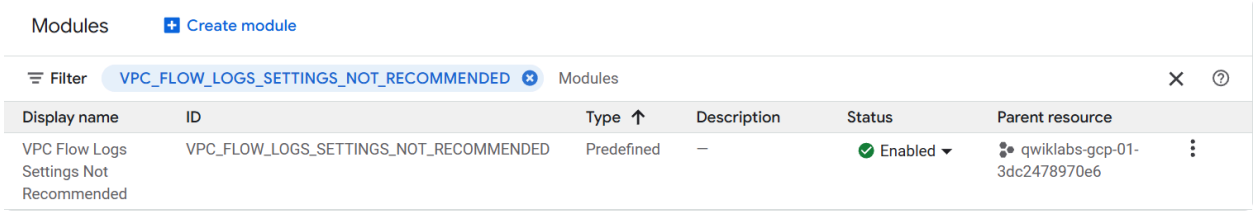



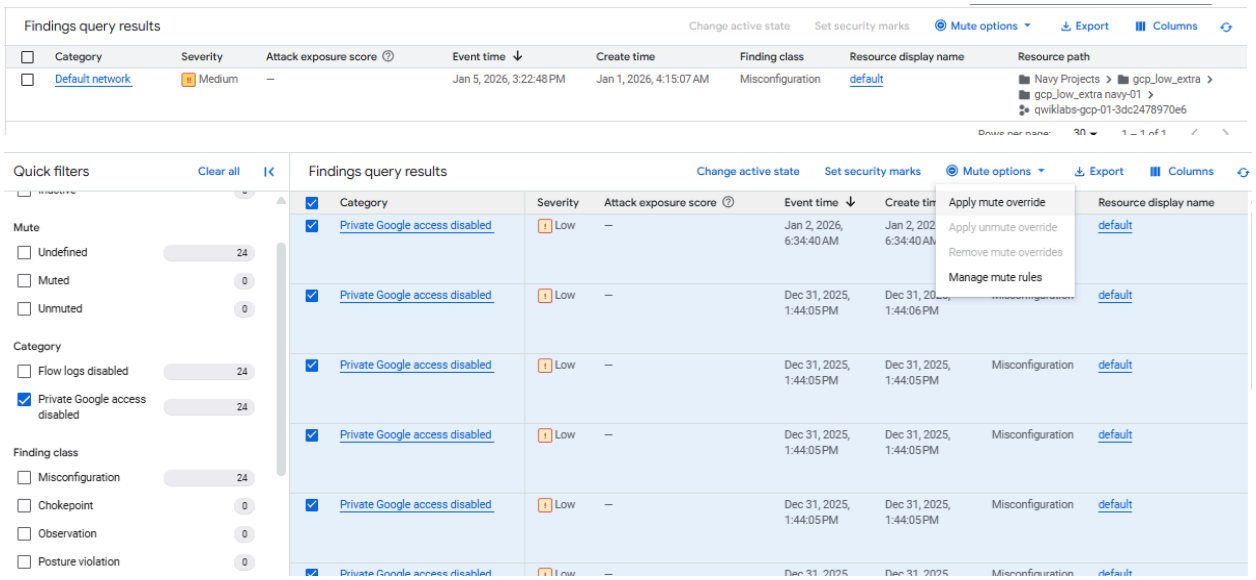



# II) Configurer les paramètres SCC au niveau du projet

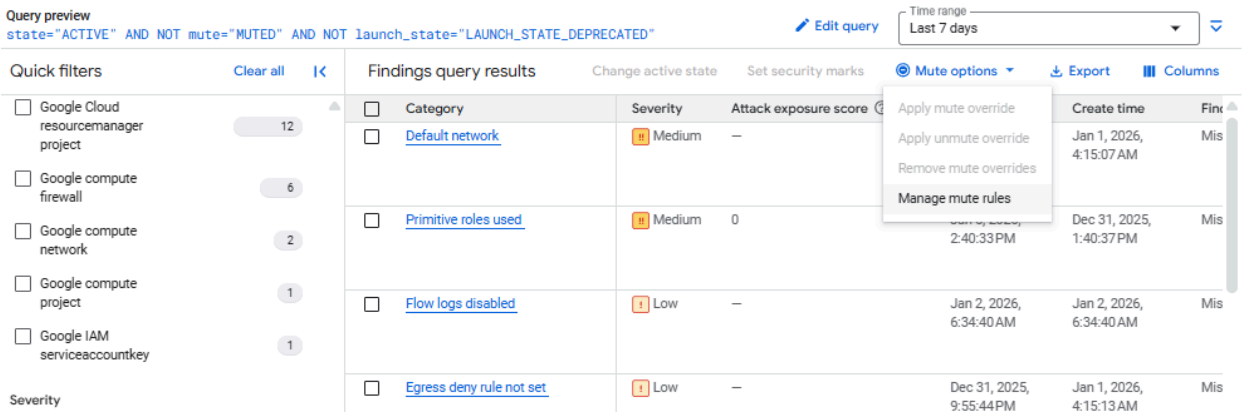


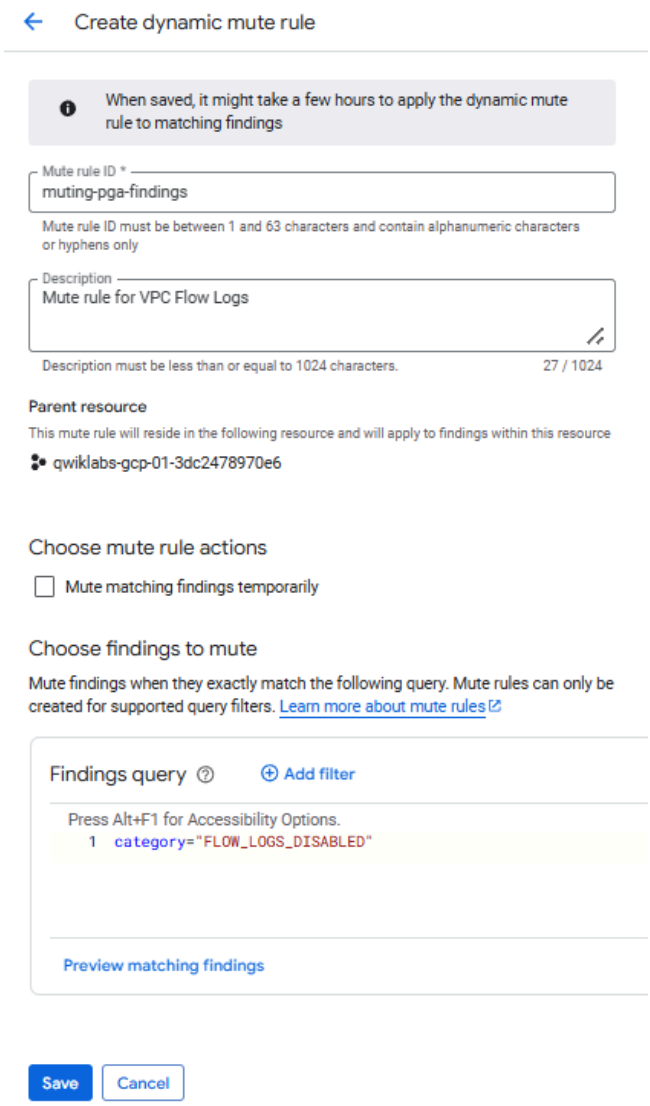

# III) Analyser et corriger les vulnérabilités SCC détectées

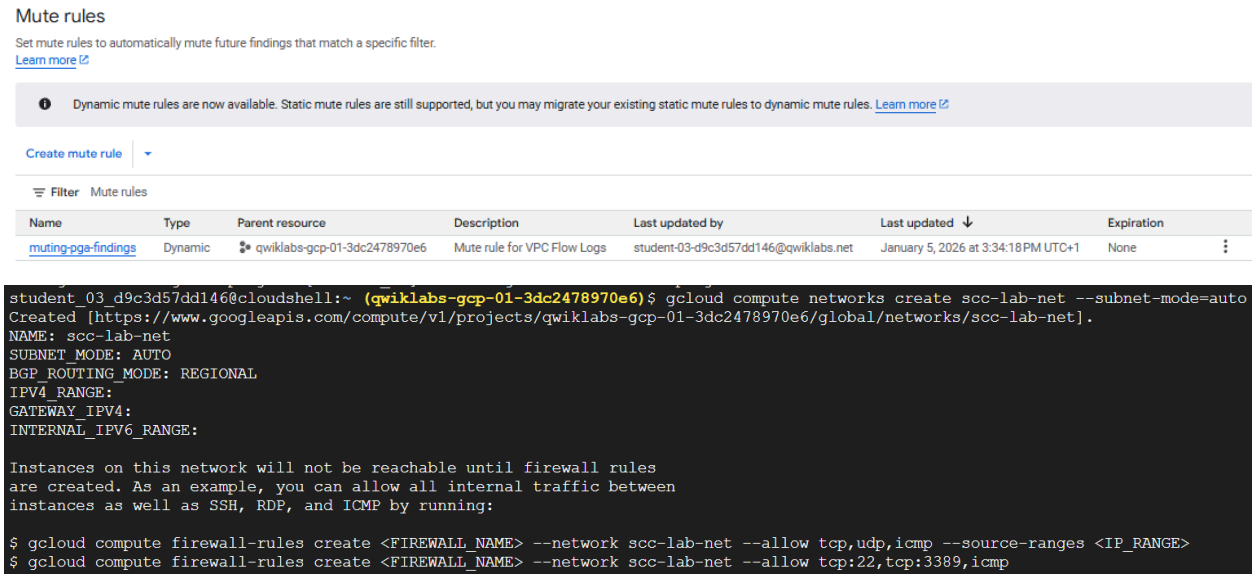



Je désactive les catégories sélectionnées avec le bouton “Apply mute override”.




Je vais dans “Manage mute rules” pour créer une règle pour désactiver les “VPC Flow Logs”.

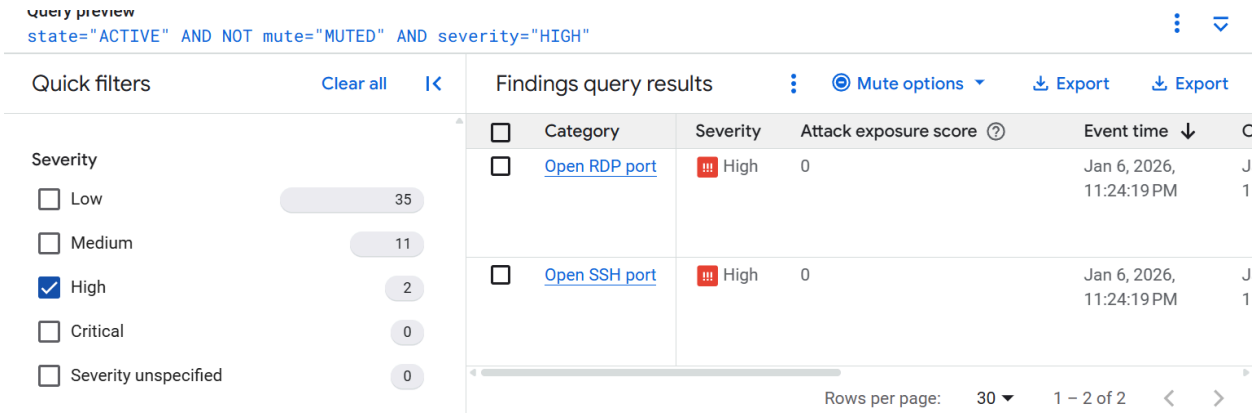




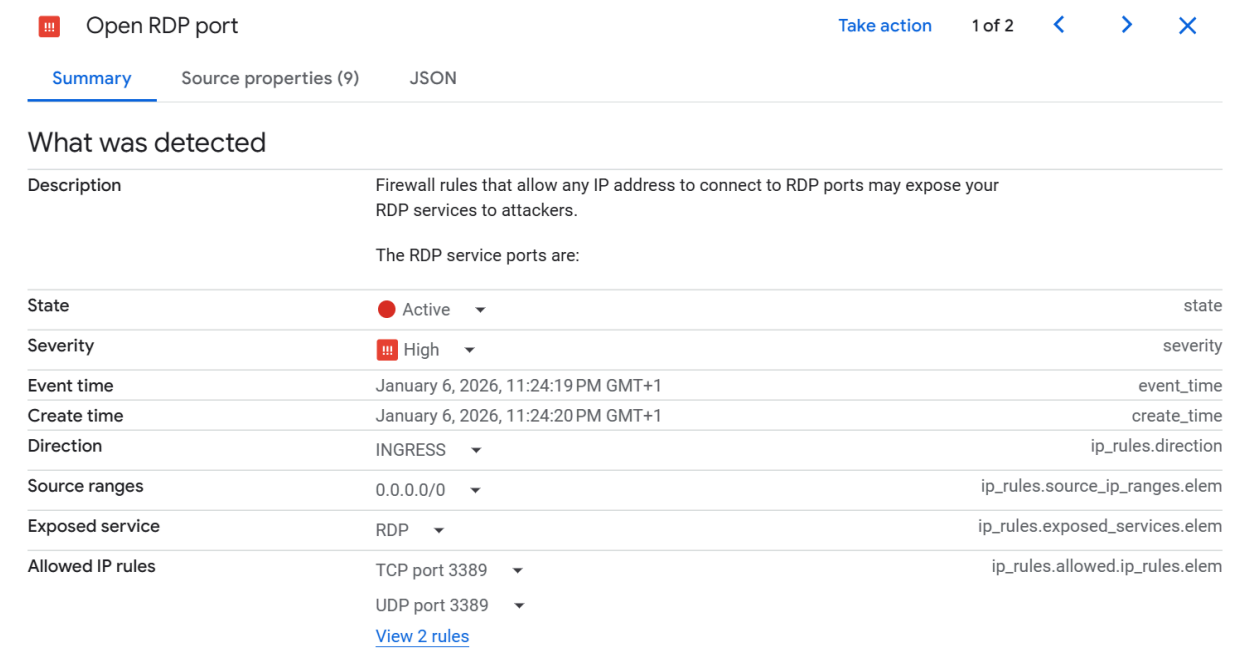

Le réseau « par défaut » contient deux règles de pare-feu qui activent le trafic Internet SSH et RDP pour toutes les instances de ce réseau.

Ainsi, j'ai deux catégories qui ont une gravité élevée :

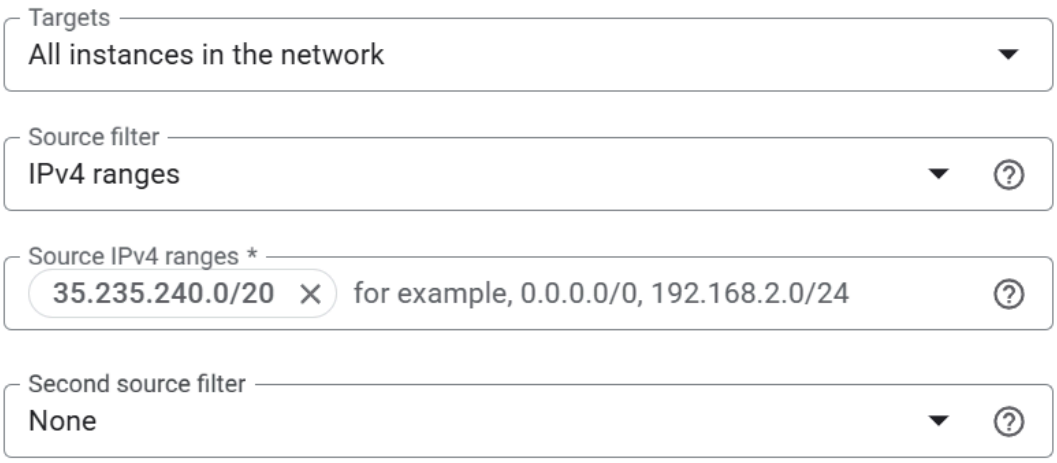
- Open RDP port
- Open SSH port

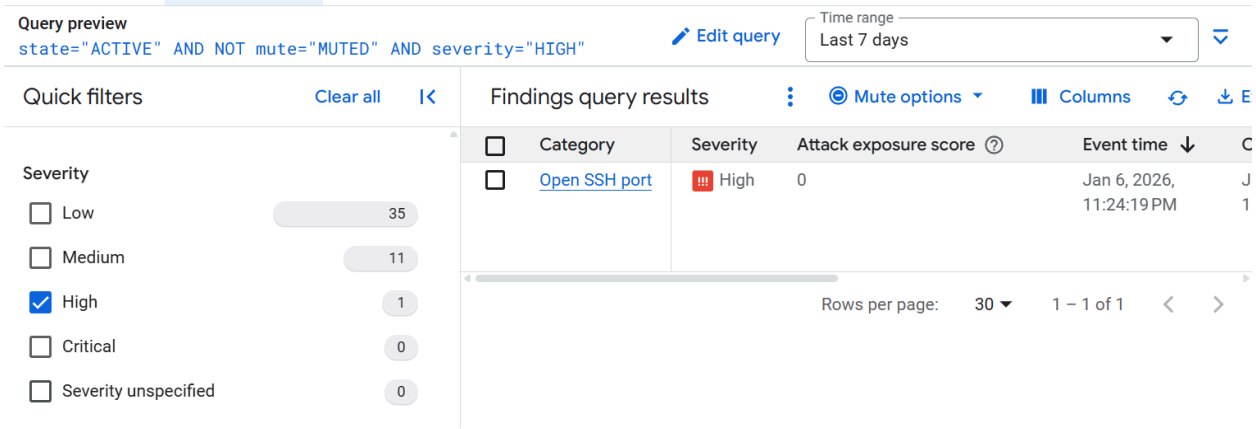



Il faut donc régler ces problèmes en modifiant les règles du pare-feu.

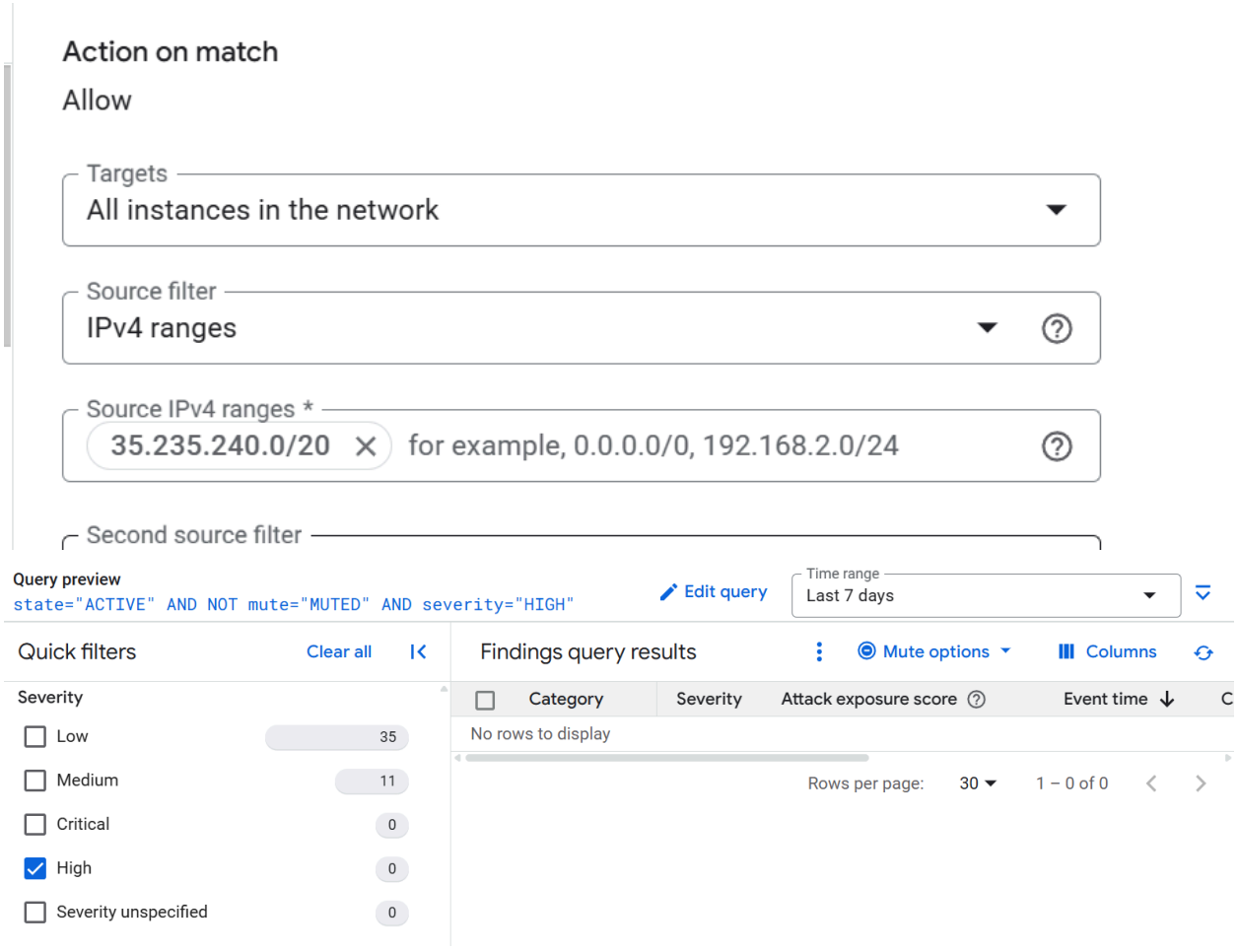



Dans la page des règles du pare-feu, j'ai modifié la plage d'adresses IP source :



De même, j'ai modifié pour SSH :



# Conclusion

Le Security Command Center (SCC) est une plateforme de surveillance de la sécurité qui aide les utilisateurs à accomplir les tâches suivantes :

- détecter les erreurs de configuration liées à la sécurité des ressources Google Cloud
- signaler les menaces actives dans les environnements Google Cloud
- corriger les vulnérabilités dans l'ensemble des ressources Google Cloud

Dans ce Google Lab, j'ai :

- exploré les éléments de l'interface SCC
- configuré les paramètres SCC au niveau du projet
- analysé et corrigé les vulnérabilités SCC détectées

| Activity | Type | Date started | Date finished | Score | Passed |
|---|---|---|---|---|---|
| Get Started with Security Command Center | Lab | 20 minutes ago | 0 minutes ago | Assessment: 100% | ✓ |
| Securing Container Builds | Lab | 1 hour ago | 25 minutes ago | Assessment: 100% | ✓ |

Tous les labs finis :