

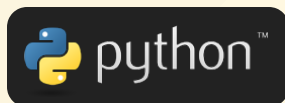
# REX analyse antivirus

des fichiers de la plateforme des emplois de l'inclusion

---

François Freitag

[mail@franek.fr](mailto:mail@franek.fr)



# Plateforme de l'inclusion



“ Faciliter la vie des **personnes en insertion** et de celles et ceux qui les accompagnent à travers de nouveaux services publics. ”

# Les emplois de l'inclusion

<https://emplois.inclusion.beta.gouv.fr>

Code open-source : <https://github.com/betagouv/itou/>

“ **Mise en relation** d'employeurs solidaires avec des candidats éloignés de l'emploi. ”

⇒ Processus de candidature : **CV** 


# Audit de sécurité

# Audit de sécurité

⚠ Pas de vérification antivirus des fichiers servis par la plateforme.



# Contraintes pour l'analyse antivirus

- 500 000+ 
- Envoyés directement sur S3 pour des raisons historiques
- Pas de **latence perceptible** à l'envoi (exigence métier)

# Quel antivirus ?



- Gratuit et open-source
- Utilisé dans d'autres start-ups d'État
- *PaaS* CleverCloud : `CC_CLAMAV=1`

# Test de performance de ClamAV

Échantillon de 10 000 fichiers aléatoires.

Temps d'analyse par fichier :

- En moyenne : 1 seconde
- Maximum : 20 secondes 🐢

**Latence perceptible** ⇒ pas d'analyse à l'envoi (requête HTTP)



# Analyse périodique 🕒

Pas de latence perceptible, mais moins de sécurité.

- **Quotidienne** des **nouveaux** fichiers
- **Mensuelle** de tous les fichiers : nouvelles signatures de virus

# Analyse périodique *a minima*

`cron` :

- Identifie les fichiers à analyser (filtre S3)
- Télécharge un lot : `TemporaryDirectory` + `ThreadPoolExecutor` ❤️
- Analyse avec ClamAV : `subprocess.run()` 😎
- Enregistre le résultat dans la base de données : *ORM* Django ❤️

# Mise en prod de la version *a minima*

- Analyse quotidienne des nouveaux fichiers
  - Parcours des objets S3 : 5 minutes
  - Analyse : 5 minutes
- Analyse mensuelle de tous les fichiers
  - Parcours des objets S3 : 5 minutes
  - Analyse : **17 280 minutes** (3 jours)
  - **SIGTERM** au déploiement (*Zero Downtime Deployment*)

# Pas très satisfaisant...

Comment éviter les interruptions liées au déploiement ? 🤔

- Pas de déploiement pendant 3 jours 🤔
- Création d'un **mécanisme de reprise**
  - Gestion du signal `SIGTERM` ⚠️ 🐉
  - Quid d'un échec sans `SIGTERM` ?
  - Acquiescement — sous quel délai ?
  - La **réponse D** : réfléchir plus...

# Analyse périodique en mieux

cron

- Identifie **mieux** les fichiers à analyser
- Télécharge un lot : `TemporaryDirectory` + `ThreadPoolExecutor` ❤️
- Analyse avec ClamAV : `subprocess.run()`
- Enregistre le résultat dans la base de données : *ORM* Django ❤️

# Préparation de l'analyse

## Une fois par jour

`cron` synchronisation S3 → base de données

## Plein de fois par jour

Sélection intelligente du lot de fichiers

- récents, ou
- dernière analyse > 1 mois

# Sélection du lot de fichiers

```
select_for_update(skip_locked=True, no_key=True)
```

# Sélection du lot de fichiers

```
select_for_update(skip_locked=True, no_key=True)
```

Préparation :

```
psql# CREATE TABLE files(id BIGINT PRIMARY KEY);  
psql# CREATE TABLE avscan(file_id BIGINT REFERENCES files (id));  
psql# INSERT INTO files VALUES (1);
```



# Sélection du lot de fichiers

```
select_for_update(skip_locked=True, no_key=True)
```

`no_key=False` :

```
psql1# BEGIN;  
psql1# SELECT * FROM files WHERE id=1 FOR UPDATE;  
psql2# BEGIN;  
psql2# INSERT INTO avscan VALUES (1);  
-- bloqué tant que psql1# n'a pas commit.
```

# Sélection du lot de fichiers

```
select_for_update(skip_locked=True, no_key=True)
```

`no_key=True` :

```
psql1# BEGIN;  
psql1# SELECT * FROM files WHERE id=1 FOR NO KEY UPDATE;  
psql2# BEGIN;  
psql2# INSERT INTO avscan VALUES (1);  
-- retourne immédiatement
```

# Qu'apporte la base de données ?

- **Mécanisme de reprise** : verrou nettoyé en cas d'échec
- Gestion de la **concurrency**
- Cerise sur le gâteau ?
  - Elle est déjà en place.

# Le résultat

624 375 fichiers scannés pour trouver...

# Le résultat

624 375 fichiers scannés pour trouver...

# Aucun virus



# Comment un virus serait traité ?

- Équipe **support** vérifie **quotidiennement** les rapports antivirus
- Confirmation que le fichier est vérolé
- Identification du **type de document** infecté
- Choix de la rémédiation

# Données d'une analyse

```
class Scan(models.Model):  
    file = models.OneToOneField(File)  
    signature = models.TextField()  
    completed_at = models.DateTimeField(null=True)  
    infected = models.BooleanField(null=True)  
    comment = models.TextField()
```

# Comment aller plus loin ?

- Stocker les fichiers via Django
- Zone de quarantaine S3
- Parallélisation des analyses
- Gestion des fichiers infectés : *API* VirusTotal



# Où voir le code ?

129 lignes :

[https://github.com/betagouv/itou/blob/master/itou/antivirus/management/commands/scan\\_s3\\_files.py](https://github.com/betagouv/itou/blob/master/itou/antivirus/management/commands/scan_s3_files.py)



# Merci

**Avez-vous des questions ?**