

# Jour-1 Tercium : Infomaniak.cloud // Paramétrage // Instanciation

**TERCIUM** : Une entreprise constituée de micro-entreprises dans divers domaines du digital. (à détailler). Il y a au total 20 personnes réalisant des projets dans divers secteurs.

- Celle-ci possède ses locaux et ses équipements en propre comme ses PC et ses NAS. Chacun étant relié aux équipes dédiées.
- Celle-ci ne stocke pas ses données dans ses propres serveurs mais les délocalisent dans une ferme de serveur Suisse qui ont une politique éthique (RSE) stricte : **Infomaniak**

**Objet du stage** : Installation de Jitsi un software de visioconférence. Ces besoins en capacité seront calculés par rapport aux besoins spécifiques.

- Website Jitsi : <https://jitsi.github.io/handbook/docs/architecture>

## Pourquoi faire des visioconférences ?

Il n'est pas toujours possible de se rendre à une réunion pour diverses raisons comme la distance ou un empêchement quelconque.

Pour faire des conversations en webcam, rencontrer plusieurs personnes, de partager des documents, des liens URL ou votre écran pour donner plus d'explications à vos interlocuteurs, il vous faudra utiliser un outil de visioconférence.

## Zoom : Standard actuel et proposition commerciale

- **Zoom Spaces** est un moyen innovant d'organiser des réunions virtuelles au sein d'équipes hybrides. Les services inclus sont la réservation d'un espace de travail (par exemple, un bureau ouvert, un dispositif Zoom), un connecteur de salle de conférence pour rejoindre les réunions avec un équipement SIP/H.323, et des salles Zoom pour les conférences qui impliquent de nombreuses personnes des deux côtés.
- **Zoom Events** est une solution dédiée à l'organisation d'événements virtuels, des annonces internes aux tables rondes. Cette option est idéale pour le réseautage, la présentation de contenu dans des stands d'exposition, la formation d'équipes dans des événements à session unique et des webinaires en ligne.
- **Zoom Contact Center** vous permet de créer un environnement omnicanal pour fournir une assistance à vos clients. Les outils disponibles comprennent un centre de contact optimisé pour la vidéo dans le cloud et l'IA conversationnelle.

## La concurrence OpenSource :

Jitsi Meet est une solution de visioconférence multiplateforme (Windows, macOS, Linux, Android, iOS) qui brille par sa modularité et son efficacité. Conçu pour les secteurs professionnels, éducatifs et médicaux, Jitsi offre une alternative open source robuste aux services payants comme Zoom ou Microsoft Teams.

**Hébergé sur un cloud multirégional, Jitsi garantit une expérience audio et vidéo fluide, même pour des utilisateurs géographiquement dispersés. Soutenue par la communauté tech et des experts en sécurité, l'application continue de se distinguer en étant entièrement gratuite et hautement personnalisable.**

**Une sécurité, qui n'est pas un détail, est l'ajout d'un mot de passe à l'appel. Cela empêche à la fois une intrusion inopportunne à une réunion ainsi que l'impossibilité à celle-ci de bloquer les appels.**

- Usage de l'icône i afin d faire apparaître les informations de la salle virtuelle de la réunion.
- Envoi via un courriel ou d'un SMS de l'URL avec le mot de passe.
- Comme zoom il suffit de placer le lien dans un navigateur ; il est recommandé d'utiliser Chrome ou Firefox.

**La capacité a évolué est permet maintenant de gérer une réunion de 500 participants. Au-delà Jitsi a un service payant : JaaS permettant une réunion de 10 000 participants.**

**Particularités :**

- **Comme ses concurrents payants anciennement Skype puis Zoom, Teams, Whereby, eyeson,,hangouts Meet ; Jitsi peut partager l'écran, enregistrer et passer en mode vignette.**
- **L'application mosaïque en est le parfait exemple.**
- **Le contrôle de sons environnement, comprendre le respect de la vie privée par l'escamotage de son espace privatif par l'usage d'un arrière-plan flou ou fictif est une fonction (usage de trois petits points).**
- **L'enregistrement de la session est possible pour une diffusion indépendante du flux avec YouTube.**
- **Celui-ci peut être aussi réaliser, l'enregistrement via Dropbox avec un compte basique gratuit, pour ce faire il faut comme précédemment utiliser les trois petits points.**

**Besoins matériels :**

- <https://jitsi.github.io/handbook/docs/devops-guide/devops-guide-requirements/>

**Usage & Compatibilité :**

- <https://linuxfr.org/users/lebouquetin/journaux/organiser-des-visioconferences-de-haute-qualite-avec-le-logiciel-libre-jitsi-meet>

**Réseau :**

- OpenRC peut être utiliser pour gérer le réseau :  
<https://www.linuxtricks.fr/wiki/openrc-gestion-du-reseau>

## Aborder Infomaniak : Cloud Provider Suisse

<https://www.infomaniak.com/fr/support/faq/2601/guide-de-demarrage-public-cloud>

Infomaniak [Public Cloud](#), une solution [Infrastructure as a service \(IaaS\)](#) basée sur [OpenStack](#) qui met à disposition les ressources dont vous avez besoin pour le développement de vos projets.

### Démarche pour une instanciation :

Inscription & accès : <https://api.pub2.infomaniak.cloud/horizon/auth/login/>

Pages explicatives : <https://docs.infomaniak.cloud/compute/>

#### Create a keypair :

CLI     Horizon

```
openstack keypair create my_keypair > ~/.ssh/my_keypair  
chmod 600 ~/.ssh/my_keypair
```

The private key is saved to `~/.ssh/my_keypair` and can then be used for instance creation (using the `--key-pair` argument on the command line) and/or for SSH.

You may now use the keypair to connect to a instance with this command :

```
ssh -i ~/.ssh/ ~/.ssh/ssh_my_rsa_key adminuser@my-instance-name
```

#### Create an instance :

##### Choose a flavor

```
$ openstack flavor list  
+-----+-----+-----+-----+-----+-----+  
| ID      | Name          | RAM | Disk | Ephemeral | VCPUs | Is Public |  
+-----+-----+-----+-----+-----+-----+  
| 21aad244-a330-4e79-ba80-4c057cf742f9 | a1-ram2-disk20-perf1 | 2048 | 20 | 0 | 1 | True |  
| 7918af3e-aa2a-4aaa-976d-9056490a4654 | a4-ram8-disk20-perf1 | 8192 | 20 | 0 | 4 | True |  
| a1d6e394-e4db-486b-8091-5d95cfb73952 | a12-ram24-disk20-perf1 | 24576 | 20 | 0 | 12 | True |  
| a35c6646-0f3c-464b-b50d-2a76cad0bd7b | a16-ram32-disk20-perf1 | 32768 | 20 | 0 | 16 | True |  
...  
| b6b7baeb-2328-48c9-8543-88cccec8ec4b | a2-ram4-disk20-perf1 | 4096 | 20 | 0 | 2 | True |  
| cd0483a8-ca2a-466b-89b2-f8d0d005408a | a8-ram16-disk20-perf1 | 16384 | 20 | 0 | 8 | True |  
+-----+-----+-----+-----+-----+-----+
```

#### Choose an image (operating System):

##### Choose an image (Operating System)

```
$ openstack image list  
+-----+-----+-----+  
| ID      | Name          | Status |  
+-----+-----+-----+  
| 097480e6-16bc-4a50-a7af-e34399d039ac | cirros-0.3.4 | active |  
| 735a5c16-56f0-4c15-80e7-49056dbc4f71 | Debian 10.10 buster | active |  
| 49f425ed-bf79-46ab-8cf3-44935d9d831e | Debian 11 bullseye | active |  
...  
| daf43e02-f59e-45bb-8d63-436094d3f360 | Ubuntu 21.04 | active |  
+-----+-----+-----+
```

## Exemple de création :

### Create your instance

```
$ openstack server create --image "Debian 11 bullseye" --flavor a2-ram4-disk20-perf1 --key-name mykeypair --network ext
+-----+-----+
| Field | Value |
+-----+-----+
| OS-DCF:diskConfig | MANUAL
| OS-EXT-AZ:availability_zone |
| OS-EXT-STS:power_state | NOSTATE
| OS-EXT-STS:task_state | scheduling
| OS-EXT-STS:vm_state | building
| OS-SRV-USG:launched_at | None
| OS-SRV-USG:terminated_at | None
| accessIPv4 |
| accessIPv6 |
| addresses |
| adminPass | mi5bBNRGRF6
| config_drive |
| created | 2021-02-24T15:51:17Z
| flavor | a2-ram4-disk20-perf1 (b6b7baeb-2328-48c9-8543-88ccce8ec4b)
| hostId |
| id | 5bf0ebf6-825d-4879-b4b8-90245ec4dc19
| image | Debian 11 bullseye
| key_name | mykeypair
| name | infomaniak-vm-1
| progress | 0
| project_id | ac4fafd60021431585bbb23470119557
| properties |
| security_groups | name='default'
| status | BUILD
| updated | 2021-02-24T15:51:17Z
| user_id | b1580497f51e4d10b9110c60c154562c
| volumes_attached |
+-----+-----+
```

## Contrôle de création d'Instance:

## Check your instance is active

```
$ openstack server show infomaniak-vm-1
+-----+-----+
| Field | Value |
+-----+-----+
| OS-DCF:diskConfig | MANUAL
| OS-EXT-AZ:availability_zone | b10
| OS-EXT-STS:power_state | Running
| OS-EXT-STS:task_state | None
| OS-EXT-STS:vm_state | active
| OS-SRV-USG:launched_at | 2021-02-24T15:51:27.000000
| OS-SRV-USG:terminated_at | None
| accessIPv4 |
| accessIPv6 |
| addresses | ext-net1=2001:1600:115:1::3d8, 195.15.242.18
| config_drive |
| created | 2021-02-24T15:51:17Z
| flavor | a2-ram4-disk20-perf1 (b6b7baeb-2328-48c9-8543-88cccec8ec4b)
| hostId | 1baedae8de146b81f259cfec3cf33fcfae980bb274f8fef46a5f49ba9
| id | 5bf0ebf6-825d-4879-b4b8-90245ec4dc19
| image | Debian 11 bullseye
| key_name | mykeypair
| name | infomaniak-vm-1
| progress | 0
| project_id | ac4fafd60021431585bbb23470119557
| properties |
| security_groups | name='default'
| status | ACTIVE
| updated | 2021-02-24T15:51:27Z
| user_id | b1580497f51e4d10b9110c60c154562c
| volumes_attached |
+-----+-----+
```

## Firewall :

### Firewall

By default, no incoming traffic is allowed to your instance but the outgoing traffic is allowed.

To allow the SSH connection you have to add a rule to the `default` security group this way :

```
openstack security group rule create --ingress --protocol tcp --dst-port 22 --ethertype IPv4 default
```

## Clef SSH : modèle

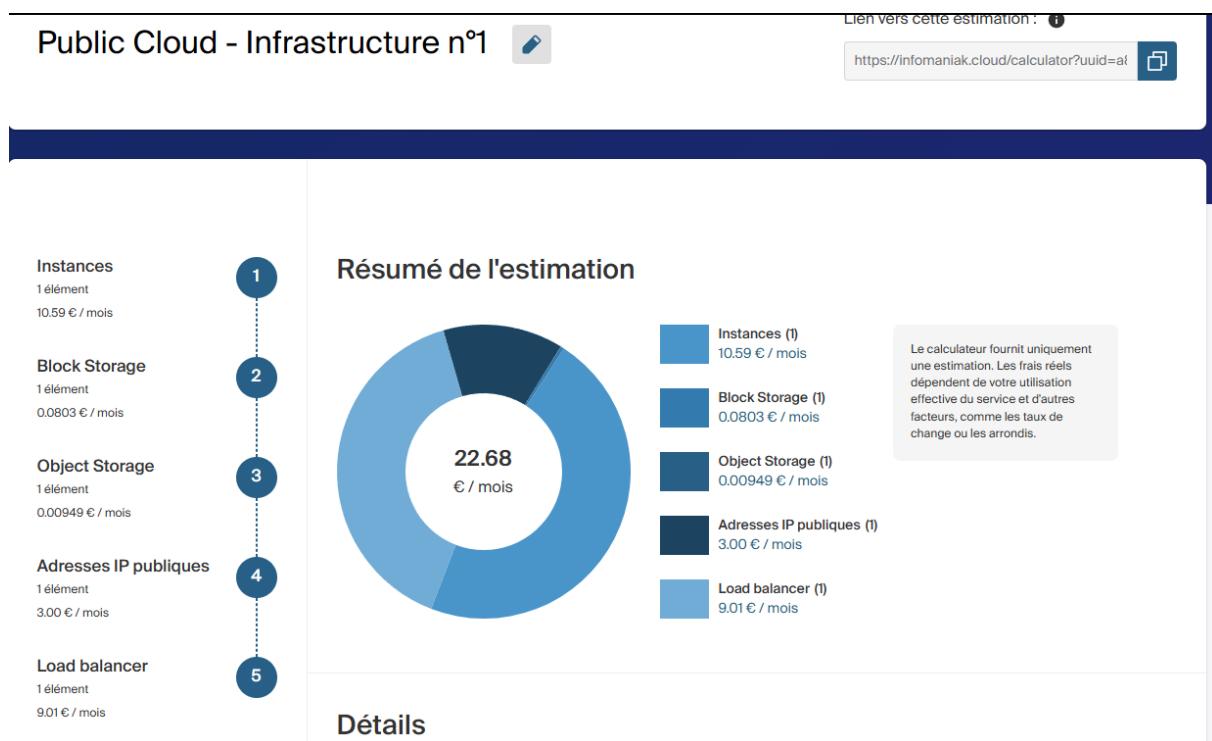
```
ssh -i path/to/your/private_key_file debian@195.15.242.18
```

## Availability Zones : openstack availability zone list

```
taylor@laptop:~$ openstack availability zone list --compute
+-----+-----+
| Zone Name | Zone Status |
+-----+-----+
| dc3-a-04 | available |
| dc3-a-09 | available |
| dc3-a-10 | available |
+-----+-----+
```

**Coût d'une instantiation pour la compagnie : Infomaniak permet la simulation.**

## Ci-dessous les choix matériels.



### Détails :

#### Instances (1)

a4_ram8_disk0	Quantité:	Temps de fonctionnement:	Système d'exploitation:	Total:
	1	730h/mois	Linux	10.59 € / mois

#### Block Storage (1)

Perf1	Quantité:	Stockage:	Total:
	1	1Go	0.0803 € / mois

#### Object Storage (1)

Object storage	Quantité:	Stockage:	Bande passante:	Total:
	1	1Go	1Go	0.00949 € / mois

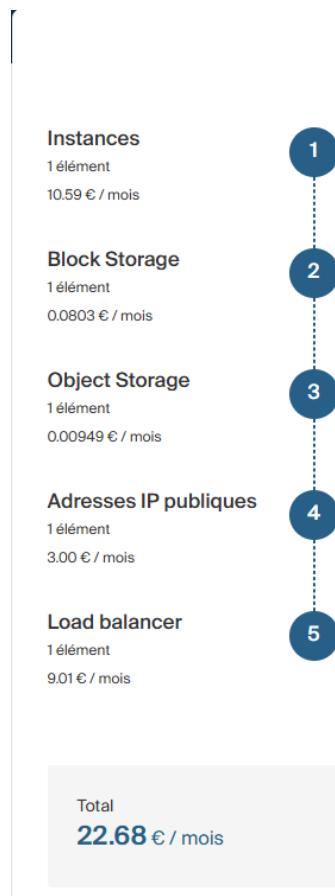
#### Adresses IP publiques (1)

IPv4 réservée	Quantité:	Total:
	1	3.00 € / mois

#### Load balancer (1)

Load balancer standard	Quantité:	Total:
	1	9.01 € / mois

### Rappel :



### Choix des availability Zone : az-1 / az-2 / az-3

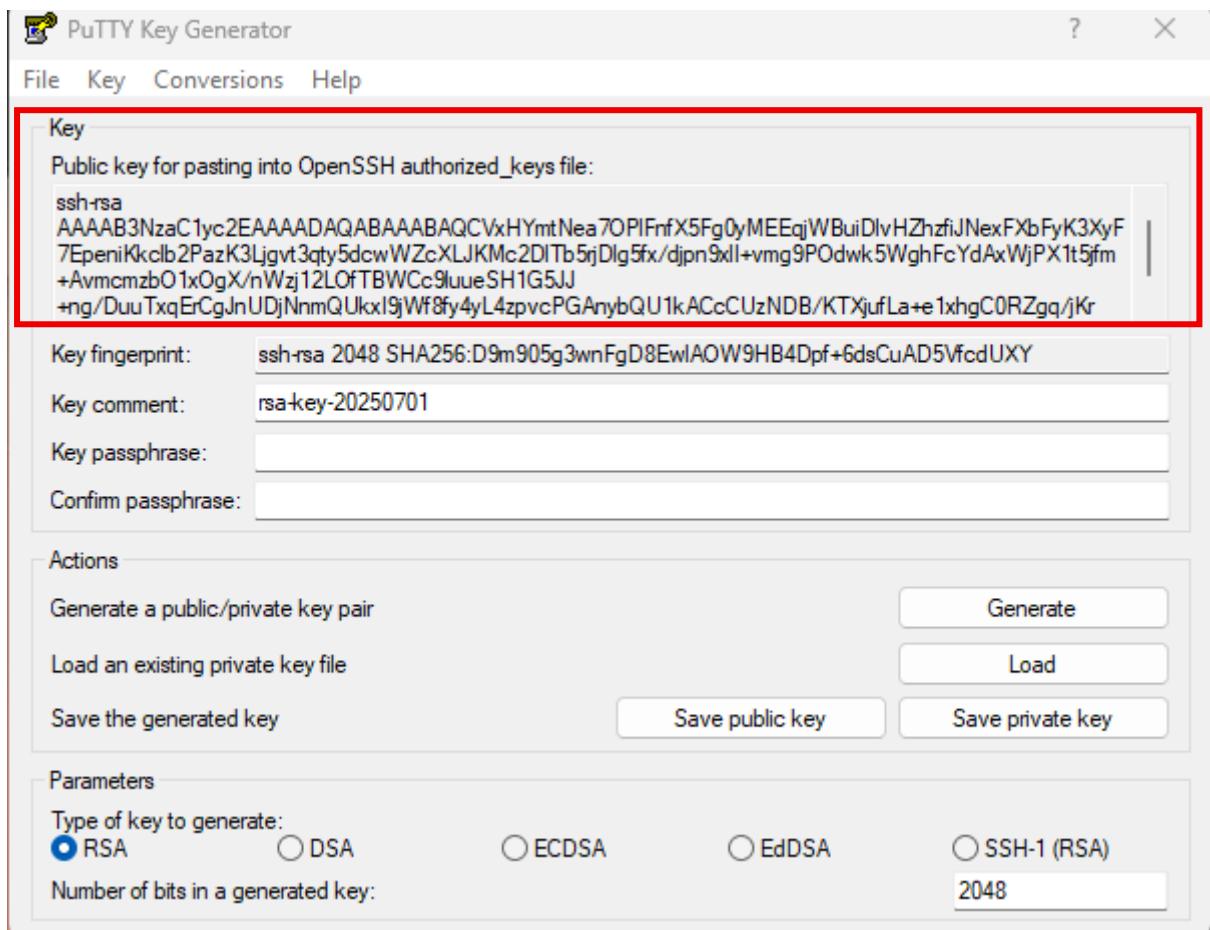
Ces trois zones sont des zones matérielles dans la ferme à serveur.

La région sera dc4-a : La région dc4-a utilisée par Infomaniak Public Cloud fait référence à un datacenter situé en Suisse, plus précisément dans la zone Genève / Satigny (canton de Genève).

**Putty Key Generator** : génère deux clefs RSA, une publique et une privée.

Je peux faire des imports dans le launch Instance fenêtre de Infomaniak.

**Putty** : me permettre en connexion, avec l'interface Putty.



Dans d'autres versions de **PuTTY** il est possible d'exporter la clef publique directement via la barre de menu, **Conversions** en format **OpenSS**.

**Ici je copie colle** la clef publique directement de **PuTTY** pour la transmettre à l'interface de création d'instance de d'infomaniak.

Dans l'impossibilité le pis allé a été de générer un fichier txt / bloc-note. Puis de l'appeler dans l'interface Infomaniak. Ce qui se déroule en deux temps.

The screenshot shows the Microsoft Azure portal interface under the 'Public Cloud' project. The 'Compute' section is selected, and the 'Key Pairs' tab is active. The table displays six key pairs:

Name	Type	Fingerprint
bonito-ssh-2025	ssh	ad:31:a0:a1:6:9d:29:de:65:7:1b:0:c2:d9:6:9
Public key	ssh	ed:b4:27:76:74:99:a3:a7:d0:a0:c4:9d:c9:0f:74:10
Public key	ssh	e6:10:c0:c6:d7:13:1a:f4:01:ab:f2:b2:dd:90:61:0f
torclum Instance_key	ssh	79:52:7c:a0:0b:a7:9c:72:5e:5c:42:d7:45:00:46:11
tercup-instance_key	ssh	79:52:7e:a0:0b:a7:9c:72:5e:5e:42:d7:45:00:45:11
test3.instance-refakt	ssh	12:96:cb:33:86:60:41:f9:e7:9c:07:3e:ce:2e:2b:b4

## Import Key Pair

X

Key Pairs are how you login to your instance after it is launched. Choose a key pair name you will recognize and paste your SSH public key into the space provided.

Key Pair Name \*

Key Type \*

Load Public Key from a file

Choose File | No file chosen

Content size: 0 bytes of 16.00 KB

Public Key \*

Cancel

Import Key Pair

## Jour-2 Tercium : Instanciation Réussie :

The screenshot shows the AWS CloudFormation Instances page. It displays two instances: 'testPublickey' and 'tercium-instance'. Both instances are in the 'Active' status, located in 'az-2' and 'az-1' availability zones respectively. The 'Actions' column for both instances has a 'Créer un instantané' button.

ID de l'instance	Filtrer	Lancer une instance	Supprimer les instances	Plus d'actions
testPublickey				
tercium-instance				

Il est plus ais   cependant de faire g  n  rer ses clefs via le terminal / console de VSCode :  
Premier probl  me l'authentification :

Il est plus ais   cependant de faire g  n  rer ses clefs via le terminal / console de VSCode :  
Premier probl  me l'authentification :

```
test@KUS-F-STAGE MINGW64 ~
$ # Cr  ez le dossier .ssh
mkdir -p ~/.ssh

# Copiez votre cl  
cp "Private_key_01_07_2025_openssh" ~/.ssh/id_rsa

# Essayez chmod (fonctionne souvent mieux dans ~/.ssh)
chmod 600 ~/.ssh/id_rsa

# V  rifiez
ls -la ~/.ssh/id_rsa
cp: cannot stat 'Private_key_01_07_2025_openssh': No such file or directory
chmod: cannot access '/c/Users/test/.ssh/id_rsa': No such file or directory
ls: cannot access '/c/Users/test/.ssh/id_rsa': No such file or directory

test@KUS-F-STAGE MINGW64 ~
$ # Retournez dans votre dossier de travail
cd "/c/Users/test/Documents/Tercium Stage"

# V  rifiez que le fichier est bien l  
ls -la Private_key_01_07_2025_openssh

# Maintenant copiez la cl   vers ~/.ssh
cp "Private_key_01_07_2025_openssh" ~/.ssh/id_rsa

# Appliquez chmod
chmod 600 ~/.ssh/id_rsa

# V  rifiez
ls -la ~/.ssh/id_rsa
-rw-r--r-- 1 test 197609 1679 Jul 1 09:15 Private_key_01_07_2025_openssh
-rw-r--r-- 1 test 197609 1679 Jul 1 10:46 /c/Users/test/.ssh/id_rsa
```

Usage de nano pour modifier les permissions :

cd : sudo nano /etc/ssh/sshd\_config Action : PasswordAuthentication no

Renforcer la s  curit   (recommand  ) :

```
bash
CopyEdit
sudo nano /etc/ssh/sshd_config
    • Modifie ou d  commente :
conf
CopyEdit
PasswordAuthentication no
PermitRootLogin prohibit-password
    • Puis red茅marre le service :
bash
CopyEdit
sudo systemctl restart ssh
```

## Via VScode je vais pouvoir faire des commandes administrateur.

- Générer et valider des droits d'usages avec chmod.
- Créer des dossiers : mkdir, des fichiers touch pour y insérer les clefs des instances.

Via le terminal Je recrée des clefs : ssh-keygen -t rsa -b 4096 -f tercium\_refait\_key qui sera le nom du fichier la contenant. Extension.pub pour la clef publique et la privée sans extension. Powershell n'étant pas le seul terminal que j'utilise, je travaille aussi avec GitBash pour avoir une ligne de commande puissante.

The terminal window shows the execution of the ssh-keygen command to generate RSA 4096 key pairs. It prompts for a passphrase and saves the keys to 'tercium\_refait'. The public key is saved as 'tercium\_refait.pub' and the private key as 'tercium\_refait'. The key fingerprint is also displayed.

```
PS C:\Users\test> ssh-keygen -t rsa -b 4096 -f tercium_refait_
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in tercium_refait_
Your public key has been saved in tercium_refait.pub
The key fingerprint is:
SHA256:KeoLkR0GvXfnYejSlduISmtGZn2/MouPaNIwLsuptI test@KUS-F-STAGE
The key's randomart image is:
+---[RSA 4096]----+
| .   o   o |
| o .  oo.* |
| = . . o = |
| + = o o .o o |
| o o o S o . |
| . . o o . o |
| . o..oo.o + |
| +..E.+o o * |
| ++oo oo. o.. |
+---[SHA256]----+
```

```
test@KUS-F-STAGE MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -f tercium_refait
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in tercium_refait
Your public key has been saved in tercium_refait.pub
The key fingerprint is:
SHA256:VMUSjInKySjCBNvGNF70bVE9Nkwliw+7jNsRSclJM test@KUS-F-STAGE
The key's randomart image is:
+---[RSA 4096]----+
| .   **=oo. |
| * o . =oX.o o |
| . + o + + BxE . |
| . . * .o.+ + |
| oS....+ |
| o . . . . |
| . . . + |
| . = o |
+---[SHA256]----+
```

## Verrouillage d'une IP :

Host terciump

HostName 37.156.45.22

User ubuntu

IdentityFile ~/Documents/Tercium\ Stage/tercium-instance\_key/tercium-instance\_key

Connexion standard : ssh -i ~/.ssh/tercium\_key [ubuntu@84.234.28.241](mailto:ubuntu@84.234.28.241)

The terminal window shows a successful SSH connection to an Ubuntu host (Ubuntu 24.04.2 LTS). The session displays system information, security status, and a prompt to run sudo apt update.

```
test@KUS-F-STAGE MINGW64 ~/Documents/Tercium Stage
$ ssh -i ~/.ssh/tercium_key ubuntu@84.234.28.241
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-68-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Tue Jul  1 09:16:31 AM UTC 2025

 System load:      0.0
 Usage of /:        20.1% of 19.52GB
 Memory usage:     3%
 Swap usage:       0%
 Processes:        158
 Users logged in:  0
 IPv4 address for enp3s0: 84.234.28.241
 IPv6 address for enp3s0: 2001:1600:16:10::7c1

 Expanded Security Maintenance for Applications is not enabled.

 0 updates can be applied immediately.

 Enable ESM Apps to receive additional future security updates.
 See https://ubuntu.com/esm or run: sudo pro status

 The list of available updates is more than a week old.
 To check for new updates run: sudo apt update

 The programs included with the Ubuntu system are free software;
 the exact distribution terms for each program are described in the
 individual files in /usr/share/doc/*copyright.

 Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
 applicable law.

 To run a command as administrator (user "root"), use "sudo <command>".
 See "man sudo_root" for details.

ubuntu@test3publickey:~$ sudo nano /etc/ssh/sshd_config
ubuntu@test3publickey:~$
```

**cd : direction instance**

**ssh -i "~/Documents/Tercium Stage/test3\_instance\_key/test3publickey\_refait\_key" ubuntu@84.234.28.241**

```
test@KUS-F-STAGE MINGW64 ~/Documents/Tercium Stage/test3_instance_key
$ ssh -i ~/Documents/Tercium Stage/test3_instance_key/test3publickey_refait_key" ubuntu@84.234.28.227
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-60-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Tue Jul 1 11:27:44 AM UTC 2025

System load: 0.12
Usage of /: 20.1% of 19.52GB
Memory usage: 2%
Swap usage: 0%
Processes: 166
Users logged in: 0
IPv4 address for enp3s0: 84.234.28.227
IPv6 address for enp3s0: 2001:1600:16:10::4bd

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo root" for details.

ubuntu@test3-instance:~$
```

Sur VsCode on sort de l'instance du terminal :**ctrl + D**

## Infomaniak.cloud : interface de contrôle des instances

### Instances

ID de l'instance = ▾	
Affichage de 2 éléments	
Instance Name	Image Name
test3Publickey	Kaas Ubuntu 2404 Kube V1.30
tercium-instance	Kaas Ubuntu 2404 Kube V1.30

**Maintenant étapes de contrôle de connexion via les Adresses IP pour un poste extérieur.  
Installation d'un serveur Apache pour Linux : LAMP**

**cd : sudo apt update && sudo apt install apache2 php libapache2-mod-php -y**

**On Active et démarre Apache :**

**cd : sudo systemctl enable apache2**

**sudo systemctl start apache2**

## Côté serveur : Apache est-il bien démarré ? sudo systemctl status apache2

```
ubuntu@tercium-instance:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
   Active: active (running) since Tue 2025-07-01 12:47:04 UTC; 27min ago
     Docs: https://httpd.apache.org/docs/2.4/
  Process: 13372 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 13376 (apache2)
    Tasks: 7 (limit: 9434)
   Memory: 11.7M (peak: 12.0M)
      CPU: 237ms
 CGroup: /system.slice/apache2.service
         ├─13376 /usr/sbin/apache2 -k start
         ├─13379 /usr/sbin/apache2 -k start
         ├─13380 /usr/sbin/apache2 -k start
         ├─13381 /usr/sbin/apache2 -k start
         ├─13382 /usr/sbin/apache2 -k start
         ├─13383 /usr/sbin/apache2 -k start
         └─13386 /usr/sbin/apache2 -k start

Jul 01 12:47:04 tercium-instance systemd[1]: Starting apache2.service - The Apache HTTP Server...
Jul 01 12:47:04 tercium-instance apachectl[13375]: AH00558: apache2: Could not reliably determine the server's fully qualified name, using 127.0.0.1 for Port 80
Jul 01 12:47:04 tercium-instance systemd[1]: Started apache2.service - The Apache HTTP Server.
lines 1-21/21 (END)
```

- **Écoute-t-il sur les bons ports ? sudo netstat -tlnp | grep :80**  
**La version moderne de netstat => sudo ss -tlnp | grep :80**

- **Côté réseau/firewall :**

Le firewall du serveur autorise-t-il le trafic sur les ports 80/443 sur infomaniak ?

**Infomaniak a des règles de sécurité/firewall à configurer dans Security Group :**

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	ICMP	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
Egress	IPv4	TCP	1 - 24	0.0.0.0/0	-	-	<button>Delete Rule</button>
Egress	IPv4	TCP	20 - 65535	0.0.0.0/0	-	-	<button>Delete Rule</button>
Egress	IPv4	UDP	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
Egress	IPv6	IPV6-ICMP	Any	::/0	-	-	<button>Delete Rule</button>
Egress	IPv6	TCP	1 - 24	::/0	-	-	<button>Delete Rule</button>
Egress	IPv6	TCP	20 - 65535	::/0	-	-	<button>Delete Rule</button>
Egress	IPv6	UDP	Any	::/0	-	-	<button>Delete Rule</button>
Ingress	IPv4	Any	Any	-	default	-	<button>Delete Rule</button>
Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	pas de port	<button>Delete Rule</button>
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	Port 22	<button>Delete Rule</button>
Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0	-	Port 80	<button>Delete Rule</button>
Ingress	IPv4	TCP	443 (HTTPS)	0.0.0.0/0	-	port 443	<button>Delete Rule</button>
Ingress	IPv4	UDP	10000	0.0.0.0/0	-	port 10000	<button>Delete Rule</button>
Ingress	IPv6	Any	Any	-	default	-	<button>Delete Rule</button>

Test simple : Créez un fichier index.html basique dans /var/www/html/ et tentez d'y accéder via <http://VOTRE IP/>

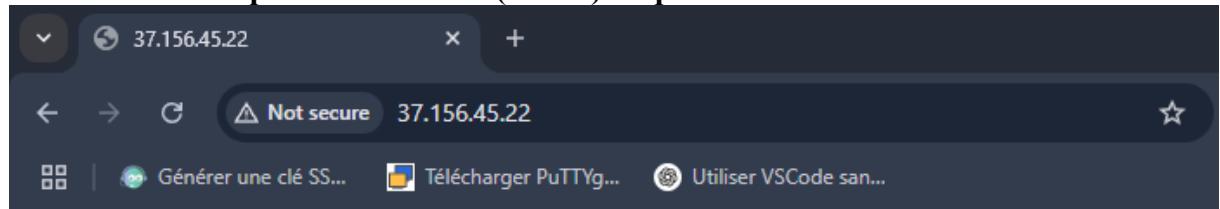
Via le terminal écrire :

```
cd : echo "<h1>OK LAMP / INFOMANIAK</h1>" | sudo tee /var/www/html/index.html
```

```
ubuntu@tercium-instance:~$ echo "<h1>OK LAMP / INFOMANIAK</h1>" | sudo tee /var/www/html/index.html
<h1>OK LAMP / INFOMANIAK</h1>
ubuntu@tercium-instance:~$ cat /var/www/html/index.html
<h1>OK LAMP / INFOMANIAK</h1>
ubuntu@tercium-instance:~$
```

Explication du code, qui fait deux choses :

1. echo produit la chaîne : "<h1>OK LAMP / INFOMANIAK</h1>"
2. tee :
  - o écrit cette chaîne dans le fichier /var/www/html/index.html
  - o et réimprime sur l'écran (stdout) ce qu'elle écrit.



## OK LAMP / INFOMANIAK

# Jour-3 Tercium : Clef SSH et Hash

## A / Fingerprint : Explication structurée issue de ChatGPT

Dans le contexte d'OpenStack (ici via l'interface *Infomaniak Public Cloud*), le fingerprint SSH associé à chaque paire de clés a les usages suivants :

### Usage du Fingerprint SSH

#### 1. Vérification d'authenticité

- Le *fingerprint* est un empreinte condensée (hash) de la clé publique associée à une paire SSH.
- Il permet de vérifier rapidement que la clé utilisée pour se connecter à une machine correspond bien à celle attendue, sans afficher toute la clé (souvent longue).

#### 2. Identification rapide des clés

- Permet de différencier deux clés ayant le même nom ou des noms similaires (ex : Public key, Public-key).
- Ex : **tercium-instance\_key** et **terciup-instance\_key** ont la même empreinte → c'est exactement la même clé.

#### 3. Sécurité (anti-usurpation)

- Lorsqu'un utilisateur ou script tente une connexion SSH, le système peut comparer le fingerprint local (de .ssh/known\_hosts) à celui enregistré dans OpenStack pour valider l'identité.

#### 4. Audit / Journalisation

- En environnement multi-utilisateur ou en audit de sécurité, l'empreinte permet de tracer quelle clé a été utilisée, sans afficher ou stocker la clé complète.

### Format de l'empreinte :

- Généralement SHA-1 (comme ici), parfois SHA-256 selon la configuration.
- Exemple : ad:31:ae:a1:76:9d:29:de:65:71:fb:07:c2:d9:6f:98

### Cas spécifique :

- **tercium-instance\_key** et **terciup-instance\_key** ont exactement le même fingerprint : Cela indique que la même clé publique a été réutilisée pour plusieurs instances.

## Bonnes pratiques

Recommandation	Description
Générer une clé unique par instance	Pour éviter des accès non maîtrisés entre VM
Stocker et vérifier les fingerprints dans un fichier de suivi	Par exemple inventory.md ou dans un dépôt Git
Comparer fingerprint local / cloud	Pour éviter les erreurs de clé en cas de multiples jeux

## B / Test de connexion simple : Ping avec le terminal sans connexion direct :

```
test@KUS-F-STAGE MINGW64 ~
$ ping 84.234.28.98

Envoi d'une requête 'Ping' 84.234.28.98 avec 32 octets de données :
Réponse de 84.234.28.98 : octets=32 temps=18 ms TTL=49
Réponse de 84.234.28.98 : octets=32 temps=16 ms TTL=49
Réponse de 84.234.28.98 : octets=32 temps=15 ms TTL=49
Réponse de 84.234.28.98 : octets=32 temps=17 ms TTL=49

Statistiques Ping pour 84.234.28.98:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perde 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 15ms, Maximum = 18ms, Moyenne = 16ms
```

## C / 2<sup>nd</sup> Test simple : Créez un fichier index.html basique dans /var/www/html/ et tentez d'y accéder via [http://VOTRE\\_IP/](http://VOTRE_IP/)

Pour se connecter via son terminal il faut activer son agent SSH avec sa clef privée :

- Se positionner dans le dossier maître :  
il se situe dans C:\Users\test\Documents\Tercium\_Stage\Tercium-instance\_key
- Puis Naviguer dans le dossier contenant les deux fichiers Public\_key & Private\_Key.pub.

**cd** : ssh -i Tercium-instance\_key/tercium-instance\_key [ubuntu@84.234.28.98](mailto:ubuntu@84.234.28.98)

si l'on oublie l'adresse IP : **Résultat**

```
test@KUS-F-STAGE MINGW64 ~
$ ssh -i /c/Users/test/Documents/Tercium_Stage/Tercium-instance_key
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface] [-b bind_address]
           [-c cipher_spec] [-D [bind_address:]port] [-E log_file]
           [-e escape_char] [-F configfile] [-I pkcs11] [-i identity_file]
           [-J destination] [-L address] [-l login_name] [-m mac_spec]
           [-O ctl_cmd] [-o option] [-P tag] [-p port] [-R address]
           [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
           destination [command [argument ...]]
ssh [-Q query_option]
```

## Résumé des processus à respecter et comprendre :

1. **Chemin correct** de la clé privée :
2. ssh -i Tercium-instance\_key/tercium-instance\_key ubuntu@84.234.28.98
3. **Clé privée avec permissions sécurisées** (si besoin) :
4. chmod 600 Tercium-instance\_key/tercium-instance\_key
5. **Pas besoin de ssh-add** dans ce cas : ssh -i suffit pour l'authentification directe.
6. **Pas de session root à maintenir ouverte pour SSL** : les certificats sont posés une fois (ex. via certbot) et renouvelés **automatiquement** en tâche de fond.

Trois éléments critiques à documenter dans le **manuel d'usage** :

### Résumé à intégrer dans ton rapport (section SSH / accès distant)

1. **Structure des fichiers :**
  - o Dossier : Tercium-instance\_key/
    - Fichier : tercium-instance\_key → **clé privée**
    - Fichier : tercium-instance\_key.pub → **clé publique**
2. **Connexion réussie via Git Bash** : ssh -i Tercium-instance\_key/tercium-instance\_key ubuntu@84.234.28.98 / à chaque instance détruite / recréer, l'**IP adresse change**.
3. **Écueils à éviter :**
  - o Ne pas confondre **répertoire** et **fichier de clé**
  - o La commande ssh-add est inutile si on utilise directement ssh -i
  - o Vérifier les permissions (chmod 600) si accès refusé
  - o Le format Windows (\) ne fonctionne pas dans Git Bash, utiliser / ou ~/Documents/...

## Proposition bonus

### Créer un alias dans Git Bash pour ne pas avoir à taper toute la ligne :

```
echo "alias ssh-tercium='ssh -i ~/Documents/Tercium_Stage/Tercium-instance_key/tercium-instance_key
ubuntu@84.234.28.98'" >> ~/.bashrc
source ~/.bashrc
```

Ensuite, il te suffira de taper : ssh-tercium

Vos pages HTML sont-elles dans /var/www/html/ ?

Ayant détruit l'instance il n'est plus possible d'obtenir quelque chose qui a été effacée.

Pour repartir proprement, séquence minimale sur Ubuntu (cloud ou local) pour reconstruire :

```
sudo apt update
sudo apt install apache2 -y
echo "<h1>Site opérationnel</h1>" | sudo tee /var/www/html/index.html
sudo systemctl restart apache2
```

## D / Test local :

- En local : <http://localhost>

Tester le serveur Apache localement (via WSL2 à installer au préalable via powershell)

**cd : wsl --list --verbose // Vérification si WSL est installé dans VSCode**

**Sinon : wsl --install -d Ubuntu**

**Aussi Installer l'extension “Remote - WSL” dans VS Code**

Extensions (Ctrl+Shift+X) → cherche : nginx  
CopyEdit  
Remote - WSL

**Lancer un projet ou un terminal dans WSL via VS Code**

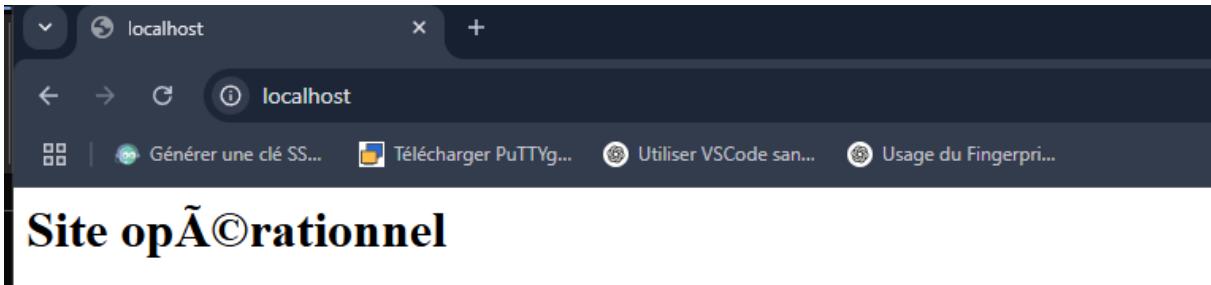
- **Ouvrir la palette de commandes** Ctrl+Shift+P
- **Tape : WSL: New WSL Window**
- **Un nouveau VS Code va s'ouvrir avec le terminal WSL actif**
- **Navigue dans le projet** (cd /mnt/c/Users/...) ou clone un dépôt

**Intérêts :**

- Ecrire depuis Windows mais exécutes sur Linux réel.
- Faire sudo, apt, systemctl, etc.
- Ouvrir des projets en un clic avec code . dans WSL

Test en local : <http://localhost> ou plus précisément <http://localhost:80>

```
test@KUS-F-STAGE:/mnt/c/Users/test$ sudo apt update
sudo apt install apache2 -y
echo "<h1>Site opérationnel</h1>" | sudo tee /var/www/html/index.html
sudo systemctl restart apache2
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
 0% [Waiting for headers]
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
120 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.58-1ubuntu8.6).
0 upgraded, 0 newly installed, 0 to remove and 120 not upgraded.
<h1>Site opérationnel</h1>
test@KUS-F-STAGE:/mnt/c/Users/test$
```



Si en cloud : `http://<ip_publique_VM>` déjà réaliser.

## E / Certificat SSL via Cerbot !

**Synthèse du document LE-SA-v1.5 (Let's Encrypt Subscriber Agreement) :**  
Daté du 24 février 2025, utile pour toute implémentation réelle ; en entreprise, audit ou projet certbot / HTTPS.

### Objectif du document :

Contrat juridique entre toi (ou ton organisation) et ISRG (Internet Security Research Group), organisme émetteur des certificats Let's Encrypt via le protocole ACME.

#### 1. Définitions clés à retenir

- **ACME** : Protocole automatisé de gestion des certificats.
- **Certificat** : Lien validé entre un nom de domaine et une clé publique.
- **Key Pair** : Paire de clés asymétriques (**privée / publique**).
- **Private Key compromise** : Clé privée compromise ou à risque => certificat à révoquer.

#### 2. Conditions d'entrée en vigueur

- **Accord effectif** dès que tu demandes un certificat **Let's Encrypt** (même via ACME).
- Il reste valide tant que tu possèdes un certificat actif, même s'il est renouvelé automatiquement.

#### 3. Engagements du souscripteur :

##### Tu garantis :

- Que tu es légitime sur le domaine visé.
- Que tu n'as pas obtenu ce domaine illégalement.
- Que tu protèges ta clé privée.
- Que les infos du certificat sont exactes, à jour et sincères.
- Que tu révoques immédiatement le certificat si :
  - la clé est compromise,
  - le domaine t'échappe,
  - ou les données sont obsolètes.

#### 4. Gestion technique

- Le certificat est généré à partir des infos envoyées par ton client ACME (e.g., certbot).
- Tu dois vérifier les infos avant usage.
- Tu as le droit d'installer le certificat uniquement sur les serveurs mentionnés dans le champ **subjectAltName**.

## **5. Usage interdit**

- Écoute active (attaque MITM)
- Interception ou redirection de trafic non autorisé
- Toute architecture facilitant la violation de la confidentialité HTTPS

## **6. Révocation**

- Tu dois révoquer un certificat via l'API ACME si :
  - clé compromise,
  - changement de domaine,
  - données fausses.
- ISRG peut révoquer sans ton accord pour :
  - usage frauduleux,
  - décision judiciaire,
  - certificat incorrect ou détourné.

## **7. Clause de non-responsabilité (ISRG)**

- Let's Encrypt est un service gratuit sans garantie contractuelle.
- ISRG décline toute responsabilité en cas :
  - de perte,
  - de poursuite,
  - de dommage technique ou juridique.

## **8. Droit applicable**

- Loi de Californie.
- Tribunal compétent : San Jose, CA.
- Aucune tierce partie n'a de droit par ce contrat.
- Limite d'action légale : 1 an.

### **En résumé pour usage réel :**

Action	Obligation
Génération des clés	En local, jamais par ISRG
Protection de la clé privée	Obligatoire (clé = confidentielle)
Vérification du certificat	Avant toute utilisation
Révocation	Immédiate en cas de doute ou perte de contrôle
Usage	Limité au domaine validé, usage HTTPS standard uniquement
Comportement interdit	MITM, reroutage de trafic, spoofing

Avec WSL en tant qu'administrateur :

cd : sudo apt update && sudo apt install cerbot python3-certbot-apache y

# Jour-4 Tercium : Coût d'une Instanciation

## A / Informations quant au coût évaluation exemple choisi : Jitsi.tercium.xyz

Coûts complets et implications techniques pour le nom de domaine jitsi.tercium.xyz, y compris la partie certification SSL/TLS. Voici la synthèse structurée, point par point :

### 1. Nom de domaine ; Jitsi.tercium.xyz

#### ► Enregistrement :

- Registrar : **Spaceship**
- Coût initial : **\$0.98 ( $\approx 0.91$  €)**
- Renouvellement : **\$10.18 annuelle**
- Transfert : **\$9.88**

Total pour 1 an (base) :  $\approx 1$  € pour l'enregistrement + SSL gratuit via Certbot (Let's Encrypt) =  $\approx 1$  € initialement, renouvellement  $\approx 10$  €

•

TLDs	New registration	Renew	Transfer
.com	 Spaceship \$9.06 <b>\$5.87</b> COM67 <a href="#">Buy domain</a>	 Cloudflare \$9.77 <a href="#">Buy domain</a>	 Cloudflare \$9.66 <b>\$8.17</b> SPSCOMTR <a href="#">Buy domain</a>
.net	 Spaceship \$11.38 <b>\$10.44</b> NET19 <a href="#">Buy domain</a>	 Spaceship \$11.38 <a href="#">Buy domain</a>	 Spaceship \$11.38 <b>\$11.3</b> SPST25 <a href="#">Buy domain</a>
.co	 Spaceship \$9.32 <b>\$6.94</b> DOM80 <a href="#">Buy domain</a>	 Spaceship \$23.98 <a href="#">Buy domain</a>	 Spaceship \$23.98 <b>\$21.18</b> SPST25 <a href="#">Buy domain</a>
.org	 Spaceship \$5.8 <b>\$5.75</b> ORG20 <a href="#">Buy domain</a>	 Spaceship \$9.98 <a href="#">Buy domain</a>	 Spaceship \$9.68 <a href="#">Buy domain</a>
.xyz	 Spaceship \$2.04 <b>\$0.98</b> XYZ52 <a href="#">Buy domain</a>	 Cloudflare \$10.18 <a href="#">Buy domain</a>	 Cloudflare \$10.57 <b>\$9.88</b> SPST25 <a href="#">Buy domain</a>
.io	 Spaceship \$38.98 <b>\$32.9</b> IO85 <a href="#">Buy domain</a>	 Sav \$38.95 <a href="#">Buy domain</a>	 Sav \$38.98 <b>\$38.88</b> SPST25 <a href="#">Buy domain</a>
.me	 Spaceship \$9.28 <b>\$1.98</b> SPSR86 <a href="#">Buy domain</a>	 Cloudflare \$14.19 <a href="#">Buy domain</a>	 Cloudflare \$15.53 <b>\$12.48</b> SPST25 <a href="#">Buy domain</a>
.info	 Spaceship \$2.77 <b>\$2.73</b> SPSR86 <a href="#">Buy domain</a>	 Dotology \$13.17 <a href="#">Buy domain</a>	 Dotology \$18.29 <b>\$13.16</b> SPST25 <a href="#">Buy domain</a>
.ai	 Spaceship \$68.98 <a href="#">Buy domain</a>	 Spaceship \$68.98 <a href="#">Buy domain</a>	 Spaceship \$68.98 <a href="#">Buy domain</a>
.top	 Spaceship \$1.23 <b>\$0.88</b> SPSR86 <a href="#">Buy domain</a>	 GoDaddy \$3.17 <a href="#">Buy domain</a>	 GoDaddy \$4.03 <b>\$3.98</b> SPST25 <a href="#">Buy domain</a>

## 2. Certificat SSL/TLS via Let's Encrypt

► Certbot avec Apache :

- Commande (déjà mentionnée dans ton journal) :

**sudo apt update**

**sudo apt install certbot python3-certbot-apache -y**

1. Configuration automatique : certbot configure Apache et obtient un certificat gratuitement via ACME.
2. Renouvellement automatique : inclus (/etc/cron.d ou via systemd timer), sans coût.

► Conditions :

- Tu dois posséder le domaine (tercium.xyz)
- Le port 80 (HTTP) doit être ouvert pour validation ACME
- Le certificat est valide 90 jours, renouvelable

## 3. Instance cloud utilisée

D'après capture :

- Type : a4-ram4-disk80-perf1
  - 4vCPU
  - 8 GB RAM
  - 20 GB disque SSD
- OS : Ubuntu 24.04 Kube v1.30 (image publique, QCOW2)  
→ Suffisant pour héberger Jitsi, mais attention au coût horaire chez Infomaniak

## 4. Configuration Apache pour Jitsi

Si tu dédies un sous-domaine jitsi.tercium.xyz :

- Ajouter un VirtualHost Apache :

```
<VirtualHost *:80>
    ServerName jitsi.tercium.xyz
    DocumentRoot /var/www/jitsi
</VirtualHost>
    • Et SSL après certbot --apache :
<VirtualHost *:443>
    ServerName jitsi.tercium.xyz
    SSLEngine on
    SSLCertificateFile /etc/letsencrypt/live/jitsi.tercium.xyz/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/jitsi.tercium.xyz/privkey.pem
</VirtualHost>
```

## 5. Checklist récapitulative

Étape	Action technique	Coût
Nom de domaine <b>tercium.xyz</b>	Achat sur Spaceship	~1 €
Création sous-domaine <b>jitsi.*</b>	Ajout dans DNS (type A vers IP publique)	0 €
Serveur Apache / installation Jitsi	Apache + Jitsi install sur Ubuntu 24.04	0 €
Ouverture port 80 / 443	Configuration Security Groups Infomaniak	0 €
Certificat TLS	Certbot + Let's Encrypt (ACME)	0 €
Renouvellement automatique SSL	Certbot cron ou systemd timer	0 €
Renouvellement domaine .xyz	Annual fee Spaceship	~10 €

### Conclusion :

- **Domaine** **tercium.xyz** : 0.98 \$ la première année
- **SSL Let's Encrypt** : 0 €
- **Infomaniak** (coût instance selon durée d'usage)
- Possibilité de pointer **jitsi.tercium.xyz** via un enregistrement **DNS** vers l'**IP publique**
- **Jitsi Meet** peut être installé et certifié gratuitement si ton reverse proxy Apache est bien configuré

### Calcul Gobal :

Coût d'une instance : Infomaniak // <https://www.infomaniak.com/fr/domaines>  
<https://hellosafe.fr/hebergeurs/infomaniak>

#### 1. Domaine réel : [visio.workeezconnect.fr](https://visio.workeezconnect.fr)

#### 2. Choix de l'OS :

➤ Kaas Ubuntu 2404 Kube V1.30	6/16/25 7:03 PM	4.21 GB	QCOW2	Public
-------------------------------	-----------------	---------	-------	--------

#### 3. Achat d'un volume / basique puis extensible.

➤ a4-ram8-disk20-perf1	4	8 GB	20 GB	20 GB	0 GB	Yes	
------------------------	---	------	-------	-------	------	-----	--

#### 4. IP fixe ou flottante : une seule adresse menant vers le serveur Jitsi

#### 5. Instance basique : moindre coût

Pour calculer le coût réel d'une instance cloud déployée comme Jitsi, surtout chez Infomaniak Public Cloud (OpenStack), tu dois prendre en compte plusieurs paramètres facturables à l'usage. Voici la méthode de calcul complète et générique, suivie de son application à ton cas (instance Jitsi jusqu'au 19 août).

## 1. Paramètres de tarification à surveiller (chez Infomaniak)

Infomaniak Public Cloud facture à la minute (via OpenStack), sur les bases suivantes :

Composant	Base de coût	Unité
vCPU	€ par vCPU par heure	€/vCPU/h
RAM	€ par GB RAM par heure	€/GB/h
Stockage disque	€ par GB par mois	€/GB/mois
Trafic sortant	€ par Go sortant par mois	€/Go/mois
Adresse IP fixe	parfois facturée séparément	€/IP/mois

## 2. Application à ton instance actuelle : Instance utilisée : a4-ram8-disk20-perf1

Ressource	Valeur
vCPU	4
RAM	8 GB
Disque	20 GB SSD
Durée d'usage	du 03/07 au 19/08 = 47 jours ≈ 1.53 mois

## Estimation du coût pour cette instance (Infomaniak Public Cloud) :

Ressource	Tarif unitaire estimé	Calcul sur 47 jours (1.53 mois)	Coût
vCPU (4)	0.01 €/vCPU/h = 0.04 €/h	$0.04 \times 24 \text{ h} \times 47 \text{ j} = 45.12 \text{ €}$	45 €
RAM (8 GB)	0.01 €/GB/h = 0.08 €/h	$0.08 \times 24 \text{ h} \times 47 \text{ j} = 90.24 \text{ €}$	90 €
Disque (20 GB)	0.05 €/GB/mois	$20 \times 0.05 \times 1.53 \text{ mois} = 1.53 \text{ €}$	1.5 €
IP publique fixe	~1.50 €/mois	$1.50 \times 1.53 = 2.30 \text{ €}$	2.3 €
Trafic sortant	Estimation : 30 Go @ 0.03 €/Go	$30 \times 0.03 = 0.90 \text{ €}$	0.9 €

### 3. Estimation du coût total pour 47 jours (1.53 mois)

Ressource	Calcul	Total approximatif
vCPU + RAM	$(0.02 + 0.04) \text{ €/h} \times 24 \text{ h} \times 47 \text{ jours} = \sim 68 \text{ €}$	68 €
Disque 80 GB	$4 \text{ €/mois} \times 1.53 \text{ mois} = 6.12 \text{ €}$	6.12 €
IP publique fixe	$1.5 \text{ €/mois} \times 1.53 \text{ mois} = 2.30 \text{ €}$	2.30 € (option)
Trafic sortant	Ex : $30 \text{ Go} \times 0.03 \text{ €/Go} = 0.90 \text{ €}$	0.90 €

→ Total estimé : 77 à 80 € pour 47 jours.

### 4. Paramètres à moniturer activement

Élément	Commande / outil (Linux Ubuntu)
Uptime	uptime ou who -b
Usage CPU/RAM	htop, top, free -h
Utilisation disque	df -h
Bandé passante	vnstat, iftop, ou Grafana/Prometheus si installé
Durée instance	Interface Infomaniak (historique / instance details)

### 5. Choix à prévoir avant le 19 août

Option	Avantage	Risque ou coût
Arrêt instance	Ne facture plus CPU/RAM, garde stockage/IP	Disque/IP facturés
Suppression instance	Zéro coût	Perte de config complète
Snapshot + arrêt	Archiver Jitsi (image QCOW2), puis stop	Très faible coût
Continuer	Service en ligne	50–80 €/mois

## Recommandation

Créer un tableau de calcul automatique avec :

- Coût horaire vCPU
- Coût horaire RAM
- Coût mensuel disque
- IP (fixe ou non)
- Trafic estimé

**B / Le 03 / 07 / 2025 Lier Adresse IP et Nom de Domaine :**

<https://www.infomaniak.com/fr/support/faq/2025/lier-un-nom-de-domaine-a-un-hebergement-web-infomaniak>

Nom de domaine fixé : <https://visio.workeezconnect.fr/>

Adresse IP fixée : **84.234.29.126**

Explication théorique claire :

Élément	Rôle
DNS (A record)	Associe visio.workeezconnect.fr → 84.234.29.126
Certificat SSL (Let's Encrypt)	Prouve que le serveur web contrôle réellement ce domaine
ACME challenge	Mécanisme utilisé pour prouver que tu possèdes le domaine

**Avant même l'installation du certificat il faut lier domaine et l'IP : Interface Infomaniak**

## Interface Infomaniak : DNS Zones Create Zone

Create Record Set

Type \*  
A - Address record

Name \*  
visio.

Description  
visioo-conférence device

TTL \*  
300

Records  
Record  
84.234.29.126

+ Add Record

x Cancel ✓ Submit



workeezconnect.fr.

Create Record Set ▾

Overview

Record Sets

### Details

ID	bf01ffa4-f9d8-4d93-a067-7b4021738474
Description	visio-conférence
Type	Primary
Status	Active
Action	None
Email	admin@workeezconnect.fr
Serial	1751537163
Time To Live	3600
Version	-

### Attributes

#### Attributes

### Modification Times

Created At	2025-07-03T09:58:50.000000
Updated At	2025-07-03T10:06:08.000000
Transferred At	-

### Associations

Pool ID	794ccc2c-d751-44fe-b57f-8894c9f5c842
Project ID	7646a67b63144f89bb5938cff2ccba60
Masters	

# workeezconnect.fr.

Create Record Set ▾

Overview Record Sets

## Details

ID	bf01ffa4-f9d8-4d93-a067-7b4021738474
Description	visio-conférence
Type	Primary
Status	Active
Action	None
Email	admin@workeezconnect.fr
Serial	1751537163
Time To Live	3600
Version	-

## Modification Times

Created At	2025-07-03T09:56:50.000000
Updated At	2025-07-03T10:08:08.000000
Transferred At	-

## Attributes

### Attributes

Pool ID	794ccc2c-d751-44fe-b57f-8894c9f5c842
Project ID	7046a67b63144f89bb5938cff2ccba60
Masters	

Project / DNS / Zones

Back

# workeezconnect.fr.

Create Record Set ▾

Overview Record Sets

Click here for filters or full text search.

Displaying 3 items

Name	Type	Records	Status	
➤ visio.workeezconnect.fr.	A - Address record	84.234.29.126	Active	<button>Update ▾</button>
➤ workeezconnect.fr.	NS - Name server	ns1.pub2.infomaniak.cloud.,ns2.pub2.infomaniak.cloud.	Active	
➤ workeezconnect.fr.	SOA - Start of authority record	ns1.pub2.infomaniak.cloud. admin.workeezconnect.fr. 1751537163 3534 600 86400 3600	Active	

Displaying 3 items

## Details des créations :

workeezconnect.fr.

Create Record Set ▾

Displaying 3 items			
Name	Type	Records	Status
visio.workeezconnect.fr.	A - Address record	84.234.29.126	Active
	Notes	None	ID 3941a16d-edab-4e93-ab9b-2976d187aa8f
	Description	visio-conférence	
workeezconnect.fr.	NS - Name server	ns1.pub2.infomaniak.cloud.,ns2.pub2.infomaniak.cloud.	Active
	Notes	None	ID 69267247-8ff5-4f80-8371-cbfc161cc354
	Description	None	
workeezconnect.fr.	SOA - Start of authority record	ns1.pub2.infomaniak.cloud. admin.workeezconnect.fr. 1751537163 3534 600 86400 3600	Active
	Notes	None	ID b7a0e55a-3048-4630-bd50-30f50f55fd1e
	Description	None	

Displaying 3 items

## Tests avec : dig workeezconnect.fr NS +short

- **dig** : commande DNS utilisée pour interroger les serveurs DNS.
- **workeezconnect.fr** : le nom de domaine que tu veux interroger.
- **NS** : signifie *Name Server*, c'est-à-dire que tu demandes les serveurs de noms responsables de ce domaine.
- **+short** : format de sortie simplifié (juste la réponse, pas les détails).

```
test@KUS-F-STAGE:/mnt/c/Windows/System32$ dig +trace visio.workeezconnect.fr
;; communications error to 10.255.255.254#53: timed out
;; communications error to 10.255.255.254#53: timed out
;; communications error to 10.255.255.254#53: timed out

; <>> Dig 9.18.30-0ubuntu0.24.04.2-Ubuntu <>> +trace visio.workeezconnect.fr
;; global options: +cmd
;; no servers could be reached
test@KUS-F-STAGE:/mnt/c/Windows/System32$ dig visio.workeezconnect.fr @ns1.pub2.infomaniak.cloud

; <>> Dig 9.18.30-0ubuntu0.24.04.2-Ubuntu <>> visio.workeezconnect.fr @ns1.pub2.infomaniak.cloud
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NXDOMAIN, id: 6766
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;visio.workeezconnect.fr.      IN      A

;; Query time: 0 msec
;; SERVER: 83.166.143.128#53(ns1.pub2.infomaniak.cloud) (UDP)
;; WHEN: Thu Jul  3 13:17:46 CEST 2025
;; MSG SIZE rcvd: 52

test@KUS-F-STAGE:/mnt/c/Windows/System32$ dig NS workeezconnect.fr +short
marjory.ns.cloudflare.com.
ajay.ns.cloudflare.com.
test@KUS-F-STAGE:/mnt/c/Windows/System32$ |
```

**Problèmes : les tests de connexions et d'identifications ont échoué, les serveurs sont des cloudfares non des infomaniak.** Il faudra changer via l'accès client permettant la création de nom de domaines pour switcher les serveurs.

### Actions requises :

Action	Où ?	Objectif
Modifier les serveurs de noms NS	Chez le registrar du domaine (OVH, Infomaniak, Gandi, etc.)	Remplacer *.ns.cloudflare.com → ns1.pub2.infomaniak.cloud
Attendre propagation DNS	Internet (1h à 24h)	Reflète les bons serveurs
Tester avec dig ou nslookup	Ton terminal	Confirmer propagation

### Vérification après propagation : avec ces commandes et les réponses attendues

1. **dig NS workeezconnect.fr +short**

# Doit répondre avec :

ns1.pub2.infomaniak.cloud.  
ns2.pub2.infomaniak.cloud.

2. **dig visio.workeezconnect.fr +short**

# Doit renvoyer :

84.234.29.126

### Délai de propagation mondial réaliste.

Niveau de cache / infrastructure	Délai estimé max
Serveurs racines & TLD (.fr)	15 à 30 min
Résolveurs publics (Google 8.8.8.8, etc.)	15 à 60 min
CDN (Cloudflare, Akamai...)	1 à 2 h
FAI (France, Europe, Afrique...)	2 à 24 h
Pays lointains (zones isolées, Asie, etc.)	24 à 48 h max

## Test : Réussi

```
ubuntu@jitsi-tercium:~$ dig +short visio.workeezconnect.fr  
84.234.29.126
```

## 2<sup>nd</sup> Test : Réussi

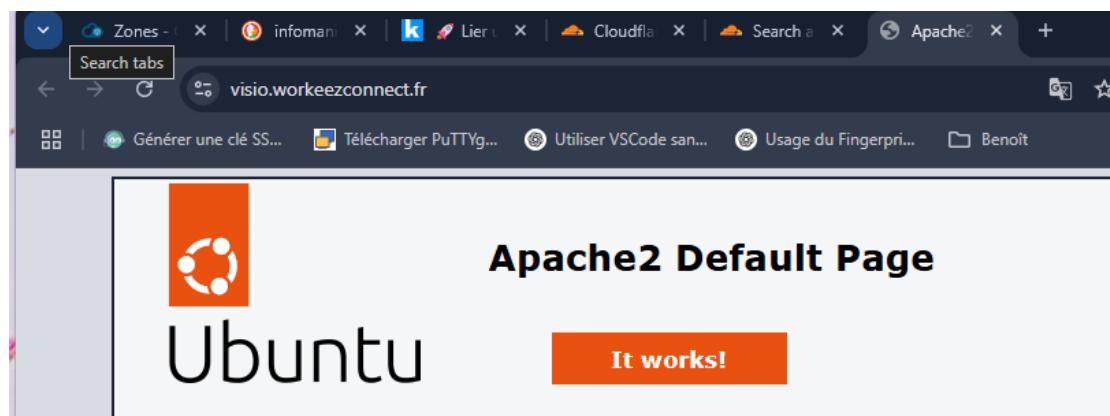
```
ubuntu@jitsi-tercium:~$ curl http://visio.workeezconnect.fr  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
  <!--  
      Modified from the Debian original for Ubuntu  
      Last updated: 2022-03-22  
      See: https://launchpad.net/bugs/1966084  
  -->
```

Cependant : le test avec le navigateur accède au site avec une non secure alert.

Relance donc de la création du certificat :

**cd : sudo certbot –apache -d visio.workeezconnect.fr**

Création réussi :



**C / Relions maintenant le serveur Jitsi à notre instance domaine + Adresse IP**

Se connecter à son dépôt via le dossier ou se situent les clefs privée et publique :  
**ssh -i Tercium-instance\_key/tercium-instance\_key [ubuntu@84.234.29.126](mailto:ubuntu@84.234.29.126)**

Ajouter le dépôt de Jitsi Meet :

**Préambule :** Le dépôt fourni par Jitsi utilise le chiffrement, donc vérifier que vous avez bien le paquet apt-transport-https.

**cd : sudo apt install apt-transport-https ca-certificates curl gnup -y**

► **apt-transport-https**

Permet à apt (le gestionnaire de paquets) de télécharger des paquets depuis des dépôts HTTPS (sécurisés via TLS). Sans cela, apt ne pourra pas accéder à <https://download.jitsi.org>.

### ► ca-certificates

Installe les autorités de certification (CA) racines utilisées pour vérifier la validité des certificats SSL/TLS lors des connexions sécurisées. Indispensable pour vérifier le certificat du dépôt Jitsi, de Let's Encrypt, etc.

### ► curl

Outil de ligne de commande pour faire des requêtes HTTP(S), utilisé ici pour récupérer la clé GPG de Jitsi.

### ► gnupg

Contient gpg (GNU Privacy Guard), utilisé pour décrypter et gérer les clés de signature GPG, comme celle du dépôt Jitsi.

### ► -y

Flag qui force l'acceptation automatique de l'installation sans demander confirmation à chaque paquet. Très utilisé en script pour éviter l'interaction manuelle.

## # Importer la clé de manière sécurisée

```
curl https://download.jitsi.org/jitsi-key.gpg.key | gpg --dearmor | sudo tee /usr/share/keyrings/jitsi-keyring.gpg > /dev/null
```

## # Ajouter le dépôt en l'associant à la clé

```
echo "deb [signed-by=/usr/share/keyrings/jitsi-keyring.gpg] https://download.jitsi.org stable/" | sudo tee /etc/apt/sources.list.d/jitsi-stable.list > /dev/null
```

## # Mettre à jour les paquets

```
sudo apt update
```

## # Installation de Jitsi Meet

```
Sudo apt install jitsi-meet -y
```

## # Associer l'IP et le nom de domaine : Déjà réaliser !

"Si votre nom de domaine pointe déjà via DNS public vers l'adresse IP de l'instance, l'étape /etc/hosts est facultative."

## # Vérifier l'état des services des serveurs

```
systemctl status jitsi-videobridge2
```

```
systemctl status jicofo
```

```
systemctl status prosody
```

## # Analyse DNS complète avec dig.

```
dig +short visio.workeezconnect.fr
```

## Jour-5 Tercium : Vérification de l'installation + derniers modules

# Vérification DNS + HTTP avec curl.  
curl -i <http://visio.workeezconnect.fr>

# Vérification croisée : IP publique actuelle du serveur.  
curl -4 ifconfig.me

Fichier à éditer : sudo nano /etc/prosody/conf.d/[visio.workeezconnect.fr.cfg.lua](http://visio.workeezconnect.fr.cfg.lua)

Code de base avec GNU nano 7.2 :

```
VirtualHost "visio.workeezconnect.fr"
    authentication = "internal_hashed"
    modules_enabled = {
        "bosh";
        "pubsub";
        "ping";
        "speakerstats";
        "conference_duration";
        "muc_lobby_rooms"; --  Ne rien ajouter après ce point-virgule dans la table !
    }
```

Component "conference.visio.workeezconnect.fr" "muc"

Component "auth.visio.workeezconnect.fr"
 authentication = "internal\_hashed"

Component "focus.visio.workeezconnect.fr"
 component\_secret = "FOCUS\_SECRET" Il faudra ajouter en sus la possibilité d'inviter

# Redémarrage Prosody :
sudo systemctl restart prosody

# Vérifier qu'il est actif :
sudo systemctl status prosody

# Vérifier si le port 5280 est ouvert :
sudo ss -tuln | grep 5280

# Tester l'accès HTTP BOSH :
curl http://localhost:5280/http-bind/

# Activer si nécessaire le Firewall sur le port ( UFW doit être actif ) :
sudo ufw allow 5280/tcp

# Vérification de la configuration
sudo prosodyctl check config

### **1/ Il manque des modules :**

```
modulemanager: Unable to load module 'conference_duration'  
modulemanager: Unable to load module 'speakerstats'  
modulemanager: Unable to load module 'muc_lobby_rooms'
```

**# On les installe avec :** sudo apt install prosody-modules && on redémarre avec sudo systemctl restart prosody

**2/ Bug car il demande un localhost (qui pose un problème lors des connexions avec l'instance). L'erreur indique que le script post-installation de jitsi-meet-prosody échoue parce qu'il attend un fichier /etc/prosody/conf.avail/localhost.cfg.lua qui n'existe pas.**

**# Création de ce fichier : localhost.cfg.lua**  
sudo nano /etc/prosody/conf.avail/localhost.cfg.lua

Contenu minimal du fichier (pour débloquer dpkg) :

```
VirtualHost "localhost"  
    authentication = "anonymous"
```

**# On relance la configuration des paquets cassés**  
sudo dpkg --configure -a

**# On vérifie l'installation**  
sudo apt install -f  
sudo systemctl restart prosody jicofo jitsi-videobridge2 nginx

**# Crée un utilisateur XMPP :**  
sudo prosodyctl register admin visio.workeezconnect.fr motdepasse

### **3/ Cas Particulier source de problèmes :**

**Conflit localhost vs nom de domaine public**

C'est le problème principal qui a perturbé Certbot / Let's Encrypt / Apache :



**Cause :**

Jitsi installe par défaut un vhost Apache localhost.conf :

```
apache  
CopyEdit  
<VirtualHost *:80>  
    ServerName localhost  
    DocumentRoot "/usr/share/jitsi-meet"  
</VirtualHost>
```

Simultanément, ton domaine réel (visio.workeezconnect.fr) est servi par :

- /etc/apache2/sites-enabled/000-default.conf (HTTP)

- /etc/apache2/sites-enabled/000-default-le-ssl.conf (HTTPS)

Certbot ou acme.sh essaie de **créer un challenge via le nom de domaine réel**, mais Apache répond par défaut avec la conf `localhost`, car elle est mal priorisée.

### ► Symptôme :

```
bash
CopyEdit
curl http://localhost/.well-known/acme-challenge/test.txt
```

donne une **redirection 301 → HTTPS sur localhost**, ce qui empêche la validation `http-01` par Let's Encrypt.

### ► Solutions recommandées :

1. **Désactiver `localhost.conf`** s'il n'est plus nécessaire :

```
bash
CopyEdit
sudo a2dissite localhost.conf
sudo systemctl reload apache2
```

2. **S'assurer que `visio.workeezconnect.fr` est le `ServerName` prioritaire** dans les fichiers `000-default.conf` et `000-default-le-ssl.conf`.
3. **Contrôler l'ordre de priorité via :**

```
bash
CopyEdit
sudo apachectl -S
```

Et **supprimer tout vhost résiduel `localhost`** qui interfère.

### ✓ Résumé logique :

Test	Commande	Interprétation
IP DNS	<code>dig +short visio.workeezconnect.fr</code>	Résultat doit être l'IP publique du serveur
IP publique (instance)	<code>curl -4 ifconfig.me</code>	Doit matcher celle obtenue par dig
Résolution + Apache OK	<code>curl -I http://visio.workeezconnect.fr</code>	HTTP 200/301/302 = OK
Conf Apache (ServerName)	<code>sudo apachectl -S</code>	Doit afficher <code>visio.workeezconnect.fr</code> en vhost

**On va installer Nginx fortement recommandé pour la sécurisation de échanges, cela implique arrêter Apache car il y aurait conflit sur le port 80 !**

**# Installer Nginx**

```
sudo apt install -y nginx
```

**# Démarrer le service**

```
sudo systemctl start nginx
```

**# Le rendre actif à chaque démarrage**

```
sudo systemctl enable nginx
```

**# Arrêt d'Apache2**

```
sudo systemctl stop apache2
```

```
sudo systemctl disable apache2
```

**# Supprimer d'Apache2**

```
sudo apt purge apache2 apache2-utils apache2-bin apache2.2-common -y
```

**Puis, s'assurer que le FQDN Fully Qualified Domain Name : nom de domaine complet incluant tous les niveaux (ex. auth.visio.workeezconnect.fr) est reconnu**

```
echo "127.0.1.1 visio.workeezconnect.fr" | sudo tee -a /etc/hosts
```

**# redémarrer Nginx**

```
sudo systemctl start nginx
```

```
sudo systemctl enable nginx
```

**# Vérifions l'état de Nginx et celui du port 80**

```
sudo systemctl status nginx
```

```
sudo lsof -i :80
```

**# Vérification les fichiers de configuration NGINX pour jitsi**

```
cat /etc/nginx/sites-available/visio.workeezconnect.fr.conf | grep ssl
```

```
cat: /etc/nginx/sites-available/visio.workeezconnect.fr.conf: No such file or directory
```

**Il faudra donc le crée**

```
sudo nano /etc/nginx/sites-available/visio.workeezconnect.fr.conf
```

```
server {
```

```
    listen 80;
```

```
    server_name visio.workeezconnect.fr;
```

```
    location / {
```

```
        proxy_pass http://localhost:8000; # ou 5000 selon ton backend
```

```
        proxy_http_version 1.1;
```

```
        proxy_set_header Upgrade $http_upgrade;
```

```
        proxy_set_header Connection "upgrade";
```

```

proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
}

location /http-bind {
    proxy_pass http://localhost:5280/http-bind;
    proxy_set_header Host $host;
    proxy_http_version 1.1;
    proxy_set_header X-Forwarded-For $remote_addr;
}
}

```

#### # Créer le lien symbolique

```
sudo ln -s /etc/nginx/sites-available/visio.workeezconnect.fr.conf /etc/nginx/sites-enabled/
```

#### # Vérifions la configuration

```
sudo nginx -t
```

#### # Redémarrer Nginx

```
sudo systemctl restart nginx
```

#### # Insérons dans ce fichier un bloc sécurisé avec HTTPS + Cerbot et vérifions que le certificat est présent

```
sudo ls /etc/letsencrypt/live/visio.workeezconnect.fr/
```

#### # Non il faut installer le plugin Certbot pour Nginx

```
sudo apt install python3-certbot-nginx -y
```

#### Puis Relancer la commande Certbot

```
sudo certbot --nginx -d visio.workeezconnect.fr
```

## 4/ Ici je stop l'installation et je débug car il y a eu trop de manipulations générant des bugs récurrents.

**1-Corriger les erreurs d'installation des paquets Jitsi (dpkg) et assurer une base fonctionnelle propre.**

**cd: hostnamectl**

```

ubuntu@visio:~$ hostnamectl
Static hostname: visio.workeezconnect.fr
Icon name: computer-vm
Chassis: VM
Machine ID: 2627f45bbb694092bcc3d701d9e436a1
Boot ID: ca67ac6e94c049c7b488d6a4eb4815a1
Virtualization: kvm
Operating System: Ubuntu 24.04.2 LTS
Kernel: Linux 6.8.0-60-generic
Architecture: x86-64
Hardware Vendor: OpenStack Foundation
Hardware Model: OpenStack Nova
Firmware Version: 1.16.3-debian-1.16.3-2-bpo12+1
Firmware Date: Tue 2014-04-01
Firmware Age: 1ly 3month 3d
ubuntu@visio:~$ 
```

## 2-Corriger /etc/hosts

**cd:** sudo nano /etc/hosts

```
GNU nano 7.2                               /etc/hosts
127.0.0.1      localhost
::1           localhost ip6-localhost ip6-loopback
127.0.1.1 visio.workeezconnect.fr auth.visio.workeezconnect.fr guest.visio.workeezconnect.fr focus.visio.workeezconnect.fr conference.visio.workeezconnect.fr
```

## 3-Redémarrer Prosody

**cd:** sudo systemctl restart prosody

## 4-Redémarrer Jicofo et JVB

**cd:** sudo systemctl restart jicofo

**cd:** sudo systemctl restart jitsi-videobridge2

## 5- Vérifier l'état

**cd:** sudo systemctl status prosody jicofo jitsi-videobridge2 **ok active (running)**

# Jour-6 Tercium 2<sup>nde</sup> semaine | Déploiement

Sur ce PC je ne suis pas en ROOT donc soit on passe en tant qu'administrateur ponctuelle : Sudo, soit passage au mode WSL.

## 1. Recréation d'une instance et modification donc de l'adresse IP.

The screenshot shows the Infomaniak Public Cloud interface. The left sidebar has sections for Project, API Access, Compute (selected), Overview, Instances (selected), Images, Key Pairs, Server Groups, Volumes, Network, Orchestration, DNS, Object Store, and Identity. The main area is titled 'Instances' and shows one item: 'Displaying 1 item'. A table lists the instance details:

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
jitsi-terci	Ubuntu 14.04 LTS (64-bit)	37.156.46.238, 2001:1600:16:10::488	a4-ram8-4 Kub	tercium-disk20-instance_key	Active	az-1	None	Running	1 hour, 43 minutes	<button>Create Snapshot</button>

## 2. Dans DNS insérer la nouvelle IP et surtout submit et contrôler celui-ci.

## 3. Pinger la nouvelle adresse IP.

```
PS C:\Users\test\Documents\Tercium_Stage> ping 37.156.46.238

Envoi d'une requête 'Ping' 37.156.46.238 avec 32 octets de données :
Réponse de 37.156.46.238 : octets=32 temps=16 ms TTL=48
Réponse de 37.156.46.238 : octets=32 temps=16 ms TTL=48
Réponse de 37.156.46.238 : octets=32 temps=15 ms TTL=48
Réponse de 37.156.46.238 : octets=32 temps=15 ms TTL=48

Statistiques Ping pour 37.156.46.238:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 15ms, Maximum = 16ms, Moyenne = 15ms
```

## 4. Commande : nouvelle instance Infomaniak : DNS

The screenshot shows the Infomaniak Public Cloud interface. The left sidebar has sections for Project, API Access, Compute, Volumes, Network, Orchestration, DNS (selected), and Identity. The main area is titled 'Zones' and shows one item: 'Displaying 3 items'. A table lists the zone records:

Name	Type	Records	Status
visio.workeezconnect.fr.	A - Address record	37.156.46.238	Active
workeezconnect.fr.	NS - Name server	ns1.pub2.infomaniak.cloud., ns2.pub2.infomaniak.cloud.	Active
workeezconnect.fr.	SOA - Start of authority record	ns1.pub2.infomaniak.cloud. admin.workeezconnect.fr. 1751879238 3534 600 88400	Active

- 5. Connexion réussie via Git Bash : ssh -i Tercium-instance\_key/tercium-instance\_key ubuntu@84.234.28.98 / à chaque instance détruite / recréer, l'IP adresse change.**

Tester avec :

**cd : dig visio.workeezconnect.fr +short**

**cd : nslookup visio.workeezconnect.fr**

**Résultat nous pointons toujours vers l'ancienne IP adress de l'instance détruite la semaine dernière.**

```
test@KUS-F-STAGE:/mnt/c/Windows/System32$ nslookup visio.workeezconnect.fr
Server:      10.255.255.254
Address:     10.255.255.254#53

Non-authoritative answer:
Name:   visio.workeezconnect.fr
Address: 84.234.29.126

test@KUS-F-STAGE:/mnt/c/Windows/System32$ dig visio.workeezconnect.fr @8.8.8.8 +short
84.234.29.126
test@KUS-F-STAGE:/mnt/c/Windows/System32$ |
```

- 6. Nous avons un conflit pour accéder aux clefs, précisément la privé de mon côté. Pour continuer nous devons alors pointer d'abord dans le bon dossier ; ici changement du nom du dossier en ssh.key:**

**En Powershell :**

**cd : C:\Users\test\Documents\Tercium\_Stage\ssh\_keys**

**En Bash (git\_bash ou en WSL) : ssh -i**

**/c/Users/test/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key -o  
IdentitiesOnly=yes ubuntu@37.156.46.238**

**Ou**

**ssh -i /c/Users/test/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key -o  
IdentitiesOnly=yes ubuntu@visio.workeezconnect.fr**

**Lors de la recréation d'une instance, L'on doit à chaque fois relier le nouvel IP Adress au domaine. Sans ce lien, il ne peut y avoir connexion SSH et d'installation de Jitsi.**

Tester avec :

**cd : dig visio.workeezconnect.fr +short**

**cd : nslookup visio.workeezconnect.fr**

```
test@KUS-F-STAGE:/mnt/c/Windows/System32$ dig visio.workeezconnect.fr @8.8.8.8 +short
37.156.46.238
test@KUS-F-STAGE:/mnt/c/Windows/System32$ nslookup visio.workeezconnect.fr
Server:      10.255.255.254
Address:     10.255.255.254#53

Non-authoritative answer:
Name:   visio.workeezconnect.fr
Address: 37.156.46.238

test@KUS-F-STAGE:/mnt/c/Windows/System32$ |
```

**cd : dig visio.workkeezconnect.fr @1.1.1.1 +trace**

```
test@KUS-F-STAGE:/mnt/c/Windows/System32$ dig visio.workkeezconnect.fr @1.1.1.1 +trace
; <>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <>> visio.workkeezconnect.fr @1.1.1.1 +trace
;; global options: +cmd
.          IN      NS      b.root-servers.net.
.          IN      NS      c.root-servers.net.
.          IN      NS      d.root-servers.net.
.          IN      NS      e.root-servers.net.
.          IN      NS      f.root-servers.net.
.          IN      NS      g.root-servers.net.
.          IN      NS      h.root-servers.net.
.          IN      NS      i.root-servers.net.
.          IN      NS      j.root-servers.net.
.          IN      NS      k.root-servers.net.
.          IN      NS      l.root-servers.net.
.          IN      NS      m.root-servers.net.
.          IN      NS      a.root-servers.net.
22627    IN      RRSIG   NS 8 0 518400 20250720050000 20250707040000 46441 . jx
DES+EpgTiUe4F8iu7MropMCs00ECSwLU73AwIoyzM EhXDSVydimmc5 dmR1U2RFxCn0ucP0sIxXgaGbfnoXAeXSasFJ99lsjYZUrVg
VFfvXg4iw COLt+1k5z7vRmwIBYLxvyW8MBkpqX/xQdRosy09VV9MGPLK8z4dp94+ Q104jEdpuat2IylWa7tvxzUZl+jjEVElc+z
SzeTMhrIGpyKNx9BpEoIU bV1h2rYWRnz/BxoNs99Hynevkv80zLsTo7z950B8KoisuNKSTym5dLf a 9UzT3XJaYg20WSijp+iyxlW
3934PCWlpTuaOvvXN3J/ieBn/eHvnCkD uWlkIg==
;; Received 525 bytes from 1.1.1.1#53(1.1.1.1) in 7 ms

;; Received 58 bytes from 202.12.27.33#53(m.root-servers.net) in 15 ms
```

On se connecte, chemin vers des fichiers clefs:

**C:\Users\test\Documents\Tercium\_Stage\ssh\_keys**

Powershell :

**cd : ssh -i "C:\Users\test\Documents\Tercium\_Stage\ssh\_keys\tercium-instance\_key" -o  
IdentitiesOnly=yes [ubuntu@37.156.46.238](mailto:ubuntu@37.156.46.238)**

Ou

Git Bash :

**cd : ssh -i ~/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key -o  
IdentitiesOnly=yes [ubuntu@37.156.46.238](mailto:ubuntu@37.156.46.238)**

Vérification complémentaires :

1/ Permissions sur la clé privée (depuis WSL/Git Bash) :

**cd : chmod 600 ~/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key**

2/ Connexion avec vérification du fingerprint :

Tu peux ajouter -v pour obtenir des logs :

**cd : ssh -i ~/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key -o  
IdentitiesOnly=yes -v [ubuntu@37.156.46.238](mailto:ubuntu@37.156.46.238)**

```

test@KUS-F-STAGE MINGW64 ~/Documents/Tercium_Stage
$ ssh -i ~/Documents/Tercium_Stage/ssh_keys/tercium-instance_key -o IdentitiesOnly=yes ubuntu@37.156.46.238
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-60-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Mon Jul 7 12:00:08 PM UTC 2025

System load:          0.0
Usage of /:           20.3% of 19.52GB
Memory usage:         3%
Swap usage:           0%
Processes:            164
Users logged in:     0
IPv4 address for enp3s0: 37.156.46.238
IPv6 address for enp3s0: 2001:1600:16:10::488

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon Jul 7 09:46:26 2025 from 90.45.118.239
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@jitsi-tercium:~$ []

```

**Nettoyer le cache DNS interne en powershell: cd : ipconfig /flushdns**

## PRÉREQUIS

1. Nom de domaine pointant vers l'IP publique : visio.workeezconnect.fr → 37.156.46.238.
2. Accès SSH root ou sudo via ta clé (Tercium-instance\_key).
3. Résolution DNS locale correcte (dig, ping, etc.).
4. Port TCP 443 (HTTPS) et 80 (HTTP) ouverts dans les règles de pare-feu GCP/Infomaniak.
5. ufw désactivé ou configuré (on vérifiera ça).
6. Pas d'Apache2 actif pour éviter conflit avec NGINX.

## ÉTAPES POUR INSTALLER JITSI + NGINX AVEC CONFIG LOCALE

### Installation des paquets requis :

```

sudo apt update && sudo apt upgrade -y
sudo apt install curl gnupg2 software-properties-common apt-transport-https
-y

```

### Ajout du dépôt Jitsi officiel :

#### Ajout la clef GPG :

```

curl https://download.jitsi.org/jitsi-key.gpg.key | sudo gpg --dearmor -o /usr/share/keyrings/jitsi-keyring.gpg

```

### Ajoute la clé GPG :

```
echo 'deb [signed-by=/usr/share/keyrings/jitsi-keyring.gpg]
https://download.jitsi.org stable/' | sudo tee
/etc/apt/sources.list.d/jitsi-stable.list
```

```
sudo apt update
```

### Installation de Jitsi Meet :

```
cd : sudo apt install -y jitsi-meet
```

💡 Pendant l'installation : Possibilités

- FQDN : visio.workeezconnect.fr
- Choisir "configurer SSL plus tard (manuel)"

## 7. Génération du certificat SSL (Let's Encrypt)

```
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d visio.workeezconnect.fr
```

Valide automatiquement le NGINX + active HTTPS.

### Étapes suivantes à vérifier pour finaliser le déploiement :

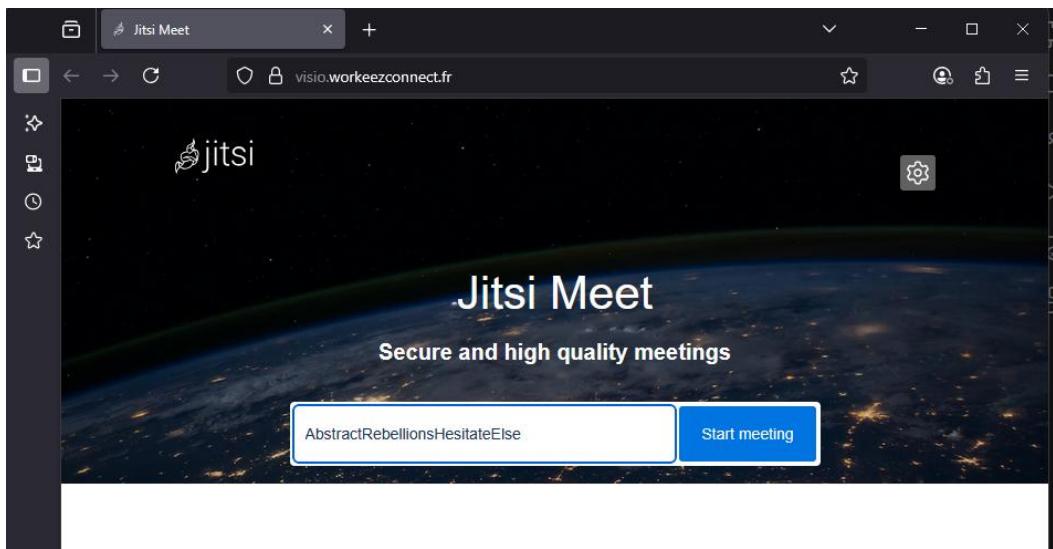
#### 1. Vérifier que Nginx est bien configuré pour Jitsi

```
cd : sudo systemctl status nginx
```

```
ubuntu@jitsi-tercium:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; pr>
   Active: active (running) since Mon 2025-07-07 12:13:02 UTC; 12min >
     Docs: man:nginx(8)
   Main PID: 48836 (nginx)
      Tasks: 5 (limit: 9434)
     Memory: 9.1M (peak: 11.2M)
        CPU: 2.475s
      CGroup: /system.slice/nginx.service
              ├─48836 nginx: master process /usr/sbin/nginx -g daemon o
              ├─61651 nginx: worker process"
              ├─61652 nginx: worker process"
              ├─61653 nginx: worker process"
              └─61654 nginx: worker process"

Jul 07 12:16:33 jitsi-tercium nginx[54327]: 2025/07/07 12:16:33 [notice]
Jul 07 12:16:33 jitsi-tercium systemd[1]: Reloaded nginx.service - A hi>
Jul 07 12:21:40 jitsi-tercium systemd[1]: Reloading nginx.service - A hi>
Jul 07 12:21:40 jitsi-tercium nginx[59285]: 2025/07/07 12:21:40 [warn] >
Jul 07 12:21:40 jitsi-tercium nginx[59285]: 2025/07/07 12:21:40 [notice]>
Jul 07 12:21:40 jitsi-tercium systemd[1]: Reloaded nginx.service - A hi>
Jul 07 12:21:52 jitsi-tercium systemd[1]: Reloading nginx.service - A hi>
Jul 07 12:21:52 jitsi-tercium nginx[61646]: 2025/07/07 12:21:52 [warn]>
Jul 07 12:21:52 jitsi-tercium nginx[61646]: 2025/07/07 12:21:52 [notice]>
Jul 07 12:21:52 jitsi-tercium systemd[1]: Reloaded nginx.service - A hi>
lines 1-25/25 (END)]
```

2. Tester la connexion HTTPS depuis un navigateur : Accède à <https://visio.workeezconnect.fr>



3. Vérifier les ports nécessaires ouverts : il faut installer ufw  
cd : sudo apt update && sudo apt install ufw

cd : sudo ufw allow 80,443/tcp  
sudo ufw allow 10000/udp  
sudo ufw enable  
sudo ufw status

```
test@KUS-F-STAGE:/mnt/c/Windows/System32$ sudo ufw allow 80,443/tcp
Rules updated
Rules updated (v6)
test@KUS-F-STAGE:/mnt/c/Windows/System32$ sudo ufw allow 1000
Rules updated
Rules updated (v6)
test@KUS-F-STAGE:/mnt/c/Windows/System32$ sudo ufw enable
Firewall is active and enabled on system startup
test@KUS-F-STAGE:/mnt/c/Windows/System32$ sudo ufw status
Status: active

To                         Action      From
--                         --          --
80,443/tcp                 ALLOW       Anywhere
1000                       ALLOW       Anywhere
80,443/tcp (v6)            ALLOW       Anywhere (v6)
1000 (v6)                  ALLOW       Anywhere (v6)

test@KUS-F-STAGE:/mnt/c/Windows/System32$ |
```

4. Vérifier les modules Prosody :  
cd : sudo cat /etc/prosody/conf.avail/visio.workeezconnect.fr.cfg.lua | grep modules\_enabled -A 20
5. Test de fonctionnement des modules : relance puis vérification de son état  
cd : sudo systemctl restart prosody  
Puis

**cd : sudo systemctl status prosody**

```
ubuntu@jitsi-tercium:~$ sudo systemctl restart prosody
ubuntu@jitsi-tercium:~$ sudo systemctl status prosody
● prosody.service - Prosody XMPP Server
   Loaded: loaded (/usr/lib/systemd/system/prosody.service; enabled; )
   Active: active (running) since Mon 2025-07-07 13:10:28 UTC; 12s ago
     Docs: https://prosody.im/doc
     Main PID: 86388 (lua5.4)
        Tasks: 1 (limit: 9434)
       Memory: 12.9M (peak: 13.6M)
          CPU: 709ms
        CGroup: /system.slice/prosody.service
                  └─86388 lua5.4 /usr/bin/prosody -F

Jul 07 13:10:28 jitsi-tercium systemd[1]: Started prosody.service - Pro
lines 1-12/12 (END)]
```

Ne pouvant contrôler Jitsi du côté d'Infomaniak, car nous ne voyons pas l'installation de celui-ci. Cependant la section DNS via Zones nous donnera ce tableau qui nous permettra de réaliser des vérifications.

1. Vérifier la résolution DNS (commande dig) :  
**cd : dig visio.workeezconnect.fr @8.8.8.8 +short**  
Réponse attendue et reçue : 37.156.48.238

```
ubuntu@jitsi-tercium:~$ dig visio.workeezconnect.fr @8.8.8.8 +short
37.156.46.238
```

2. Tester la connectivité HTTP(S) (commande curl) :  
**cd : curl -I <https://visio.workeezconnect.fr>**

```
ubuntu@jitsi-tercium:~$ curl -I https://visio.workeezconnect.fr
HTTP/2 200
server: nginx/1.24.0 (Ubuntu)
date: Mon, 07 Jul 2025 13:33:00 GMT
content-type: text/html
vary: Accept-Encoding
strict-transport-security: max-age=63072000
```

3. S'il y avait échec avec HTTPS, tester HTTP:  
**cd : curl -I <http://visio.workeezconnect.fr>**

```
ubuntu@jitsi-tercium:~$ curl -I http://visio.workeezconnect.fr
HTTP/1.1 301 Moved Permanently
Server: nginx/1.24.0 (Ubuntu)
Date: Mon, 07 Jul 2025 13:34:56 GMT
Content-Type: text/html
Content-Length: 178
Connection: keep-alive
Location: https://visio.workeezconnect.fr/
```

4. Tester le port 443 avec telnet ou nc

**cd : nc -zv visio.workeezconnect.fr 443**

```
ubuntu@jitsi-tercium:~$ nc -zv visio.workeezconnect.fr 443
Connection to visio.workeezconnect.fr (37.156.46.238) 443 port [tcp/https] succeeded!
```

5. Tester la chaîne de certificat SSL (Certbot utilisé)

**cd : echo | openssl s\_client -connect visio.workeezconnect.fr:443 -servername visio.workeezconnect.fr**

```
ubuntu@jitsi-tercium:~$ echo | openssl s_client -connect visio.workeezconnect.fr:443
servername visio.workeezconnect.fr
CONNECTED(00000003)
depth=2 C = US, O = Internet Security Research Group, CN = ISRG Root X1
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = E6
verify return:1
depth=0 CN = visio.workeezconnect.fr
verify return:1
```

- ISRG Root X1 → racine de Let's Encrypt
- Let's Encrypt E6 → intermédiaire
- visio.workeezconnect.fr → certificat de ton instance

Vérification SSL :

```
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 256 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
```

**Verify return code: 0 (ok)**

Vérification TLS :

```
Post-Handshake New Session Ticket arrived:
SSL-Session:
Protocol : TLSv1.3
Cipher   : TLS_AES_256_GCM_SHA384
Session-ID: 1133B848101D1A2A3E76A7/AE8E076BEE87764C960945D561BE09F481529A70C
Session-ID-ctx:
Resumption PSK: DF4416E07C370CB47A55B42865FE58FF06F62D0D4A0202376E0BFE1037F14504E
FBC9E0FC28A5FABA592EAFA4B0E7E582
PSK identity: None
PSK identity hint: None
SRP username: None
TLS session ticket lifetime hint: 86400 (seconds)
TLS session ticket:
0000 - 0a 19 f2 1a 1f b3 53 bb-db c6 83 ee 0a 98 28 bf  .....S.....(.  
0010 - 14 24 cf 0d ce 72 a4 11-e6 58 16 cc 2c 72 40 dc  .$.r...X.,r@.

Start Time: 1751895664
Timeout  : 7200 (sec)
Verify return code: 0 (ok)
Extended master secret: no
Max Early Data: 0
```

Conclusion :

Élément vérifié	État
Certificat SSL bien installé	✓ OK
Chaîne complète avec intermédiaire & racine	✓ OK
Nom de domaine valide (CN = visio...)	✓ OK
Protocole TLS récent (1.3)	✓ OK

## Jour-7 Tercium 2<sup>nde</sup> semaine | installation modules de monitoring

## **Monitoring Infomaniak :**

### **Si vous arrêtez temporairement l'instance (sans la supprimer) :**

- Jitsi reste installé sur le disque
- Vous devrez juste vous reconnecter en SSH avec la nouvelle adresse IP
- Aucune réinstallation nécessaire

### **Si vous supprimez complètement l'instance :**

- Tout est effacé, y compris Jitsi
- Vous devrez tout réinstaller depuis zéro sur une nouvelle instance

### **Si votre instance redémarre avec une nouvelle IP :**

- Jitsi restera installé mais il faudra probablement ajuster la configuration
- Le certificat SSL devra être régénéré si vous utilisez un nom de domaine
- Vous devrez mettre à jour l'adresse IP dans votre configuration SSH

Dans la plupart des cas avec les services cloud, un simple arrêt/redémarrage conserve vos données. Seule la suppression définitive de l'instance efface tout.

Quel est votre cas exactement ? Vous utilisez quel service cloud (AWS, Google Cloud, etc.) ?

Avec Infomaniak, vous pouvez arrêter votre instance sans l'effacer. Voici comment procéder :

### **Dans le manager Infomaniak :**

1. Connectez-vous à votre espace client Infomaniak
2. Allez dans la section "Public Cloud" ou "Serveurs"
3. Sélectionnez votre instance
4. Cliquez sur "Arrêter" (bouton stop/pause)

### **Ce qui se passe quand vous arrêtez :**

- L'instance s'arrête mais le disque virtuel est conservé
- Vous ne payez plus les ressources CPU/RAM (seulement le stockage)
- Toutes vos données et installations (dont Jitsi) restent intactes

### **Pour redémarrer plus tard :**

- Cliquez sur "Démarrer" dans le manager
- L'instance redémarre avec la même configuration
- Vous devrez peut-être noter la nouvelle adresse IP publique
- Connectez-vous en SSH avec cette nouvelle IP

**Important :** Ne cliquez pas sur "Supprimer" ou "Détruire" - cela effacerait définitivement tout.

L'arrêt d'instance chez Infomaniak est réversible et conserve vos données, contrairement à la suppression qui est définitive.

### **Planning structuré de l'audit:**

- Audit technique général
- Inventaire des environnements matériels (serveurs, réseau, postes...)
- Demande d'autorisation pour les scans réseau (très important !)
- Validation de la topologie réseau réelle

### **Concernant les autorisations :**

- netstat : généralement OK (analyse locale)
- nmap : nécessite autorisation écrite (scan réseau)
- arpscan : idem, peut être perçu comme intrusif

### **Conseils pour la demande :**

- Préciser que c'est pour valider la documentation existante
- Mentionner les créneaux horaires prévus
- Expliquer que c'est pour la sécurité/conformité
- Proposer un rapport de synthèse

### **Phase discrète :**

- Netstat poste par poste → pas d'impact réseau, juste analyse locale
- Collecte d'infos sans perturber l'activité

### **Phase avec autorisation :**

- Nmap et arpscan après accord du boss
- Approche transparente et professionnelle

**Le plus intéressant :** La configuration sécurité avec API et alertes smartphone ! C'est du monitoring intelligent :

- Alertes discrètes en temps réel
- Le boss reste informé sans être submergé
- Réactivité en cas d'incident

### **Conseils pour convaincre :**

- Expliquer que nmap/arpscan permettront de valider que cette API de sécurité ne rate rien
- Montrer que c'est pour renforcer son système d'alertes existant
- Proposer un test sur une plage horaire définie

Travail d'auditeur ! Vous combinez discréction, professionnalisme et valeur ajoutée. L'entreprise appréciera sûrement cette approche qui respecte ses processus tout en améliorant la sécurité.

Le système d'alertes smartphone montre qu'il y a une sensibilisation à la sécurité.

Déploiement et correction des bugs : pop-ups absent, absence de micro, de caméra etc.

Résumé des actions à vérifier côté client

Élément	Vérification
Pop-up caméra/micro	via cadenas navigateur
Notifications autorisées	dem
Console JS (erreurs)	check getUserMedia, ICE
Headers Nginx	Permissions-Policy, CORS
Services Jitsi	jicofo, videobridge

09h20 fin des tests :

- L'import d'un Micro et d'une caméra externe sont supportés.
- Ceux sont pour la caméra une 720p essential B et pour le micro un Razer omnidirectionnel semi-professionnel.
- Les fonctions de partage d'écran, d'invitation via un lien simple ou multiple fonctionnent.
- L'ensemble des fonctions attendues existantes sur zoom et anciennement skype sont présentes.

<https://www.onlyoffice.com/blog/fr/2023/02/zoom-ou-jitsi>

En Annexe des saisies d'écran de l'interface avec les fonctionnalités disponibles.

Comparaison à titre indicatif de l'usage de Zoom :

- **Zoom Spaces** est un moyen innovant d'organiser des réunions virtuelles au sein d'équipes hybrides. Les services inclus sont la réservation d'un espace de travail (par exemple, un bureau ouvert, un dispositif Zoom), un connecteur de salle de conférence pour rejoindre les réunions avec un équipement SIP/H.323, et des salles Zoom pour les conférences qui impliquent de nombreuses personnes des deux côtés.
- **Zoom Events** est une solution dédiée à l'organisation d'événements virtuels, des annonces internes aux tables rondes. Cette option est idéale pour le réseautage, la présentation de contenu dans des stands d'exposition, la formation d'équipes dans des événements à session unique et des webinaires en ligne.
- **Zoom Contact Center** vous permet de créer un environnement omnicanal pour fournir une assistance à vos clients. Les outils disponibles comprennent un centre de contact optimisé pour la vidéo dans le cloud et l'IA conversationnelle.
- 

#### Coût & Résultat d'usage :

Comme chaque solution comprend différents ensembles de fonctionnalités, les prix varient en fonction de votre demande.

## **Plans tarifaires :**

**Le coût d'un logiciel de vidéoconférence est un aspect essentiel pour la plupart des équipes, car chaque entreprise souhaite économiser sur les solutions numériques et investir de l'argent dans la croissance.**

**Zoom offre un plan gratuit avec les caractéristiques suivantes :**

- **Jusqu'à 40 minutes par réunion**
- **100 participants par réunion**
- **3 tableaux modifiables avec 25 Mo de stockage en nuage**
- **Chat d'équipe**

**Le plan de base est suffisant pour la communication personnelle, tandis que les entreprises envisageront des plans payants avec des fonctionnalités puissantes. Les prix commencent à 149 \$ par utilisateur/mois et peuvent inclure :**

- **Jusqu'à 1000 participants**
- **Réunions illimitées**
- **30 heures par réunion**
- **Stockage illimité des enregistrements sur le cloud**
- **Tableaux blancs illimités**
- **Authentification unique (Single Sign-On – SSO)**
- **Domaines gérés**
- **Marque de l'entreprise**

## **Sécurité :**

**La sécurité et la confidentialité sont essentielles au succès de la vidéoconférence. Personne ne souhaite que des personnes non autorisées aient accès à des discussions privées contenant des informations sensibles.**

**Jitsi** est axé sur la sécurité et propose des salles de réunion éphémères, ce qui signifie qu'elles n'existent que pendant la réunion. Les autres options de confidentialité comprennent la protection par mot de passe, le lobby, le cryptage de bout en bout.

**Jitsi Meet** pour un déploiement sur site ne dispose pas de moteurs d'analyse préconfigurés. Même dans la version en ligne, il ne conserve aucune information personnelle, comme les noms ou les adresses électroniques.

**Zoom** est considéré comme relativement sûr grâce à son cryptage de bout en bout. Toutefois, des sources fiables signalent que **Zoom utilise l'algorithme AES-128 au lieu de AES-256, ce qui rend sa sécurité plus vulnérable** dans une certaine mesure. En outre, vous ne serez pas en mesure de vérifier le code source de ce logiciel pour vous assurer qu'il ne contient pas de virus ou de portes dérobées.

**Dans la pratique, les deux solutions se sont avérées bien protégées. Si vous suivez les directives de base en matière de sécurité de l'information sur votre lieu de travail, les piratages et les attaques extérieures sont peu probables.**

**Plugin Jitsi pour ONLYOFFICE Docs : <https://www.onlyoffice.com/blog?p=26614>**

## Monitoring : 3<sup>ième</sup> étape de la gestion de projet / Explications du processus

### OBJECTIF

1. Monitorer l'instance **Jitsi** (CPU, RAM, disques, etc.).
2. Visualiser dans **Grafana**.
3. Superviser la disponibilité du service Jitsi via **Prometheus** (ou Blackbox Exporter).
4. Intégrer PfSense et ipatable.
5. Alerter si nécessaire.

### Les modes de connexion :

1/Connexion avec vérification du fingerprint : Vscode en Git Bash

Tu peux ajouter -v pour obtenir des logs :

```
cd : ssh -i ~/Documents/Tercium_Stage/ssh_keys/tercium-instance_key -o  
IdentityOnly=yes -v ubuntu@37.156.46.238
```

Naviguer via WSL demande une réécriture. Pour accéder au dossier C:\Users\test\Documents\Tercium\_Stage depuis un terminal WSL, vous devrez naviguer en utilisant le chemin Linux approprié, car WSL monte les disques Windows sous /mnt/c/.

```
cd : cd /mnt/c/Users/test/Documents/Tercium_Stage
```

Puis

```
cd : ssh -i /mnt/c/Users/test/Documents/Tercium_Stage/ssh_keys/tercium-instance_key -o  
IdentityOnly=yes -v ubuntu@37.156.46.238
```

Cependant l'accès à une instance infomaniak via WSL est bloquée. La raison les permissions SSH sont trop souples. La commande ci-dessous est refusée.

```
cd : chmod 600 /mnt/c/Users/test/Documents/Tercium_Stage/ssh_keys/tercium-  
instance_key
```

### Explication : Différences entre Git Bash et WSL

#### 1. Git Bash et les permissions de fichiers :

- **Git Bash** (sous Windows) fonctionne généralement en émulation Unix, mais il a une gestion un peu plus souple des permissions de fichiers. Il peut donc permettre l'accès à des clés privées même avec des permissions un peu plus larges.

#### 2. WSL et les permissions sous Windows :

- **Sous WSL**, les permissions de fichiers sont plus strictes, car elles sont basées sur le système de fichiers Linux qui s'exécute sur Windows. Lorsque vous

accédez à un fichier Windows via /mnt/c/, WSL essaie de respecter les permissions de fichier Linux. Par défaut, les fichiers sous /mnt/c/ peuvent être considérés comme trop accessibles si leurs permissions sont trop larges.

**Testons sous le terminal Git Bash :**

**cd** : ls -l /c/Users/test/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key

```
test@KUS-F-STAGE MINGW64 ~/Documents/tercium_Stage
$ ls -l /c/Users/test/Documents/Tercium_Stage/ssh_keys/tercium-instance_key
-rw-r--r-- 1 test 197609 3381 Jul 1 13:42 /c/Users/test/Documents/Tercium_Stage/ssh_
keys/tercium-instance_key
```

- Permissions actuelles : **rw-r--r--**
  - Propriétaire (test) : lecture et écriture (**rw-**).
  - Groupe et autres utilisateurs : lecture seule (**r--**).
- Propriétaire : test (probablement votre utilisateur actuel).
- Taille : 3381 octets.
- Date : Le fichier a été modifié pour la dernière fois le 1er juillet.

**Problème :**

Les permissions actuelles du fichier clé (**rw-r--r--**) ne sont pas suffisantes pour que SSH fonctionne correctement. SSH exige que la clé privée soit accessible uniquement par le propriétaire du fichier, ce qui nécessite des permissions strictes de type **600**.

**Modifions les permissions** pour les rendre compatibles avec SSH :

**cd** : chmod 600 /c/Users/test/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key

**2/ Connexion avec WSL :**

**Testons sous le termina WSL :**

**cd** : ls -l /mnt/c/Users/test/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key

```
test@KUS-F-STAGE:/mnt/c/Users/test/Documents/Tercium_Stage$ ls -ld /mnt/c/Users/test/Documents/Tercium_Stage/ssh_keys
drwxrwxrwx 1 test test 512 Jul 1 13:42 /mnt/c/Users/test/Documents/Tercium_Stage/ssh_keys
test@KUS-F-STAGE:/mnt/c/Users/test/Documents/Tercium_Stage$ |
```

**Modifions les permissions** pour les rendre compatibles avec SSH :

**cd** : sudo chmod 600 /mnt/c/Users/test/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key

- Des solutions et explications seront en annexe

**3/ Connexion avec Powershell :**

**Testons sous le terminal sous Powershell :**

```
cd : ssh -i C:\Users\test\Documents\Tercium_Stage\ssh_keys\tercium-instance_key -o  
IdentitiesOnly=yes -v ubuntu@37.156.46.238
```

### Restriction de privilége :

Usage possible d'icacls, un utilitaire en ligne de commande de Windows, pour modifier les permissions sur le fichier clé privée. La commande suivante restreint l'accès à la clé privée aux seuls utilisateurs autorisés :

```
cd : icacls "tercium-instance_key" /inheritance:r /grant:r "test:(R)"
```

### Explication :

- /inheritance:r : Supprime l'héritage des permissions (cela empêchera les autres utilisateurs d'hériter des permissions du répertoire parent).
- /grant:r "test:(R)" : Donne à l'utilisateur **test** (remplacez par votre nom d'utilisateur si nécessaire) l'accès en lecture seule (R) sur ce fichier.

### Vérification :

```
cd : icacls "tercium-instance_key"
```

## L'installation des outils de supervision :

L'on part toujours du dossier où se situe des deux clefs, la publique et la privée.

### 1/ Powershell :

```
cd : test@KUS-F-STAGE MINGW64 ~/Documents/tercium_Stage
```

### Connexion SSH en Powershell:

```
cd : ssh -i C:\Users\test\Documents\Tercium_Stage\ssh_keys\tercium-instance_key -o  
IdentitiesOnly=yes -v ubuntu@37.156.46.238
```

### 2/ Connexion SSH en Git Bash :

```
cd : ssh -i ~/.ssh/tercium-instance_key ubuntu@37.156.46.238 en linux ou WSL  
ssh      -i      "/c/Users/test/Documents/Tercium_Stage/ssh_keys/tercium-instance_key"  
ubuntu@37.156.46.238
```

## Installation de PROMETHEUS:

### Télécharger :

```
cd : cd ~wget  
https://github.com/prometheus/prometheus/releases/download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz  
tar -xvf prometheus-2.52.0.linux-amd64.tar.gz
```

```
sudo mv prometheus-2.52.0.linux-amd64 /opt/prometheus
```

### Service :

```
cd : sudo tee /etc/systemd/system/prometheus.service > /dev/null <<EOF
```

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target
```

```
[Service]
User=prometheus
Group=prometheus
Restart=on-failure
ExecStart=/opt/prometheus/prometheus \
--config.file=/etc/prometheus/prometheus.yml \
--storage.tsdb.path=/var/lib/prometheus \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries
--web.listen-address=0.0.0.0:9091
```

```
[Install]
WantedBy=multi-user.target
EOF
```

```
sudo systemctl daemon-reload
sudo systemctl enable --now prometheus
```

Définir les droits :

```
cd :
sudo useradd -rs /bin/false prometheus
sudo mkdir -p /etc/prometheus /var/lib/prometheus
sudo cp /opt/prometheus/prometheus.yml /etc/prometheus/
sudo cp -r /opt/prometheus/consoles /opt/prometheus/console_libraries /etc/prometheus/
sudo chown -R prometheus: /etc/prometheus /var/lib/prometheus
```

Créer le dossier de données :

```
cd :
sudo mkdir -p /var/lib/prometheus
sudo chown -R prometheus: /var/lib/prometheus
```

Recharger systemd :

```
cd : sudo systemctl daemon-reload
```

Lancer Prometheus :

```
cd : sudo systemctl enable --now prometheus
```

Vérifier :

```
cd : sudo systemctl status prometheus
curl http://localhost:9090
```

```

ubuntu@jitsi-tercium:~$ sudo systemctl daemon-reload
sudo systemctl restart prometheus
sudo systemctl status prometheus
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; preset: enabled)
   Active: active (running) since Wed 2025-07-09 08:32:30 UTC; 24ms ago
     Main PID: 258337 (prometheus)
        Tasks: 7 (limit: 9434)
       Memory: 8.6M (peak: 8.8M)
          CPU: 26ms
         CGroup: /system.slice/prometheus.service
             └─258337 /opt/prometheus/prometheus --config.file=/opt/prometheus/prometheus.yml --storage.tsdb.path=/v>

Jul 09 08:32:30 jitsi-tercium systemd[1]: Started prometheus.service - Prometheus.
1 lines 1-11/11 (END\|)

```

**Attention conflit avec Jjtsi\_meet au port 9090 passage de prometheus au port 9091 via le le service group de infomaniak et au changement dans son script voir au-dessus.**

```

ubuntu@jitsi-tercium:~$ # Vérifie l'écoute
sudo ss -tulnp | grep 9091

# Vérifie la réponse HTTP
curl http://localhost:9091

# Vérifie l'accessibilité depuis l'extérieur
curl http://[IP_PUBLIQUE]:9091
tcp  LISTEN 0      4096                           *:9091           *:*    users:(("prometheus",pid=258337
,fd=7))
<a href="/graph">Found</a>.

curl: (3) bad range in URL position 9:
http://[IP_PUBLIQUE]:9091
^
ubuntu@jitsi-tercium:~$ 

```

## Jour-8 Tercium 2<sup>nde</sup> semaine | monitoring installations

Connexion SSH en Git Bash :

```
cd : ssh -i "/c/Users/test/Documents/Tercium_Stage/ssh_keys/tercium-instance_key"
ubuntu@37.156.46.238
```

### **Installation de NODE EXPORTER :**

Télécharger :

```
cd : cd ~wget
```

```
https://github.com/prometheus/node\_exporter/releases/download/v1.8.1/node\_exporter-1.8.1.linux-amd64.tar.gz
```

```
tar -xvf node_exporter-1.8.1.linux-amd64.tar.gz
```

```
sudo mv node_exporter-1.8.1.linux-amd64/node_exporter /usr/local/bin/
```

Utilisateur & service :

```
cd : sudo useradd -rs /bin/false node_exporter
```

```
sudo tee /etc/systemd/system/node_exporter.service > /dev/null <<EOF
```

```
[Unit]
```

```
Description=Node Exporter
```

```
After=network.target
```

```
[Service]
```

```
User=node_exporter
```

```
ExecStart=/usr/local/bin/node_exporter
```

```
[Install]
```

```
WantedBy=default.target
```

```
EOF
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl enable --now node_exporter
```

### **Installation de BLACKBOX EXPORTER :**

Télécharger :

```
cd : cd / opt
```

```
sudo wget
```

```
https://github.com/prometheus/blackbox\_exporter/releases/download/v0.25.0/blackbox\_exporter-0.25.0.linux-amd64.tar.gz
```

```
sudo tar -xvf blackbox_exporter-0.25.0.linux-amd64.tar.gz
```

```
sudo mv blackbox_exporter-0.25.0.linux-amd64 /opt/blackbox_exporter
```

Utilisateur & service :

```
cd : sudo nano /etc/systemd/system/blackbox_exporter.service
```

```
[Unit]
```

```
Description=Blackbox Exporter
```

```
Wants=network-online.target
```

```

After=network-online.target
[Service]
User=prometheus
ExecStart=/opt/blackbox_exporter/blackbox_exporter \
--config.file=/opt/blackbox_exporter/blackbox.yml

[Install]
WantedBy=multi-user.target

```

**Redémarrer :**

```

sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable blackbox_exporter
sudo systemctl start blackbox_exporter

```

## Installation de GRAFANA :

**Installation via dépôt officiel :**

```

cd :          # Installation officielle (clé + repo)
sudo apt-get install -y software-properties-common
sudo apt-get install -y gnupg2 curl

sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://apt.grafana.com/gpg.key | sudo gpg --dearmor -o
/etc/apt/keyrings/grafana.gpg

echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable
main" | sudo tee /etc/apt/sources.list.d/grafana.list

sudo apt update
sudo apt install grafana -y

# Activation et lancement
sudo systemctl enable grafana-server
sudo systemctl start grafana-server

```

**Vérification avec :**

**cd : sudo systemctl status grafana-server**

```

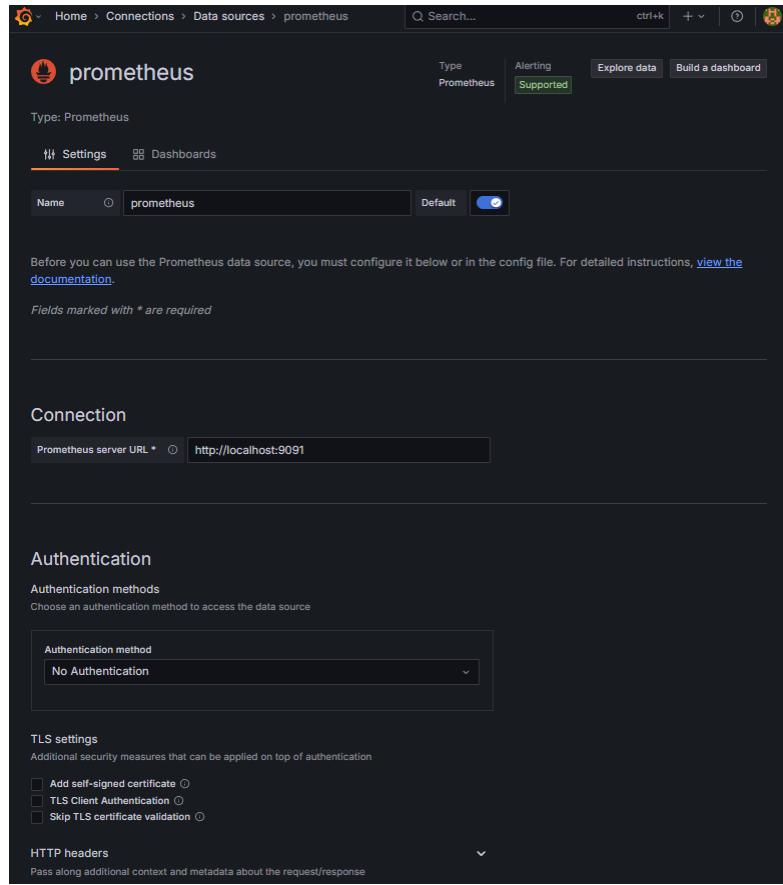
ubuntu@jitsi-tercium:~$ sudo systemctl enable --now grafana-server
Synchronizing state of grafana-server.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable grafana-server

```

● Interface Web : [http://<ip\\_publique>:3000](http://<ip_publique>:3000)

Dans notre cas cela sera : <http://37.156.46.238:3000>

Identifiant par défaut : admin / admin ici admin /BZTwwtq1964



## Installation de PFSENSE :

### 1/ Commandes pour des dépôt github bash non accessibles :

**cd :** chmod +x pfsense\_exporter-linux-amd64

**cd :** sudo mv pfsense\_exporter-linux-amd64 /usr/local/bin/pfsense\_exporter

### 2/ Télécharger et placer le binaire : sous forme de python script via un repo GitHub (sviskontree/pfsense\_exporter).

#### Contient deux fichiers :

Script principal :

**cd :** wget

[https://raw.githubusercontent.com/sviskontree/pfsense\\_exporter/master/pfsense\\_exporter.py](https://raw.githubusercontent.com/sviskontree/pfsense_exporter/master/pfsense_exporter.py)

Script secondaire de lancement :

**cd :** wget

[https://raw.githubusercontent.com/sviskontree/pfsense\\_exporter/master/pfsense\\_exporter.sh](https://raw.githubusercontent.com/sviskontree/pfsense_exporter/master/pfsense_exporter.sh)

```

ubuntu@jitsi-tercium:~$ wget https://raw.githubusercontent.com/sviskontree/pfsense_exporter/master/pfsense_exporter.py
wget https://raw.githubusercontent.com/sviskontree/pfsense_exporter/master/pfsense_exporter.sh
--2025-07-11 12:03:07-- https://raw.githubusercontent.com/sviskontree/pfsense_exporter/master/pfsense_exporter.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 2606:50c0:8002::154, 2606:50c0:8003::154, 2606:50c0:8000::154, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|2606:50c0:8002::154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1772 (1.7K) [text/plain]
Saving to: 'pfsense_exporter.py'

pfsense_exporter.py          100%[=====]  1.73K  --.-KB/s   in 0s

2025-07-11 12:03:07 (14.5 MB/s) - 'pfsense_exporter.py' saved [1772/1772]

--2025-07-11 12:03:07-- https://raw.githubusercontent.com/sviskontree/pfsense_exporter/master/pfsense_exporter.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 2606:50c0:8003::154, 2606:50c0:8000::154, 2606:50c0:8001::154, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|2606:50c0:8003::154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 748 [text/plain]
Saving to: 'pfsense_exporter.sh'

pfsense_exporter.sh          100%[=====]  748  --.-KB/s   in 0s

2025-07-11 12:03:08 (51.9 MB/s) - 'pfsense_exporter.sh' saved [748/748]

ubuntu@jitsi-tercium:~$ 

```

## Attention les scripts de pfsense sont en FreeBSD

**cd :** sudo mv pfsense\_exporter.py /usr/local/bin/  
 sudo mv pfsense\_exporter.sh /usr/local/etc/rc.d/  
 chmod +x /usr/local/bin/pfsense\_exporter.py /usr/local/etc/rc.d/pfsense\_exporter.sh

Ces commandes ne peuvent fonctionner que sous FreeBSD et sous pfsense

Il faudra créer un dossier local des services:

Donc :

**cd :** sudo mkdir -p /opt/pfsense\_exporter

```

ubuntu@jitsi-tercium:~$ sudo mkdir -p /opt/pfsense_exporter
ubuntu@jitsi-tercium:~$ ls -l
total 112644
drwxr-xr-x 2 ubuntu ubuntu    4096 Jul  8 14:07 node_exporter-1.8.1.linux-amd64
-rw-rw-r-- 1 ubuntu ubuntu 10672684 May 21 2024 node_exporter-1.8.1.linux-amd64.tar.gz
-rw-rw-r-- 1 ubuntu ubuntu     1772 Jul 11 12:28 pfsense_exporter.py
-rw-rw-r-- 1 ubuntu ubuntu      748 Jul 11 12:28 pfsense_exporter.sh
-rw-rw-r-- 1 ubuntu ubuntu 104659766 May  8 2024 prometheus-2.52.0.linux-amd64.tar.gz

```

Puis déplacer les deux fichiers avec les autorisations vers l'instance et vérifier l'export :

**cd :** sudo mv pfsense\_exporter.py /opt/pfsense\_exporter/  
 sudo mv pfsense\_exporter.sh /opt/pfsense\_exporter/  
 sudo chmod +x /opt/pfsense\_exporter/\*

**cd : ls -l**

```
ubuntu@jitsi-tercium:~$ sudo mv pfSense_exporter.py /opt/pfSense_exporter/
sudo mv pfSense_exporter.sh /opt/pfSense_exporter/
sudo chmod +x /opt/pfSense_exporter/*
ubuntu@jitsi-tercium:~$ ls -l
total 112636
drwxr-xr-x 2 ubuntu ubuntu      4096 Jul  8 14:07 node_exporter-1.8.1.linux-amd64
-rw-rw-r-- 1 ubuntu ubuntu 10672684 May 21 2024 node_exporter-1.8.1.linux-amd64.tar.gz
-rw-rw-r-- 1 ubuntu ubuntu 104659766 May  8 2024 prometheus-2.52.0.linux-amd64.tar.gz
```

**Puis vérifier l'export :**

**cd : ls -l /opt/pfSense\_exporter/**

```
ubuntu@jitsi-tercium:~$ ls -l /opt/pfSense_exporter/
total 8
-rwxrwxr-x 1 ubuntu ubuntu 1772 Jul 11 12:28 pfSense_exporter.py
-rwxrwxr-x 1 ubuntu ubuntu   748 Jul 11 12:28 pfSense_exporter.sh
```

/opt/pfSense\_exporter/  
└── pfSense\_exporter.py  
└── pfSense\_exporter.sh

**Créer un service systemd :**

**cd : sudo nano /etc/systemd/system/pfSense\_exporter.service**

**Contenu :**

**cd :**

**[Unit]**

**Description=**PfSense Exporter for Prometheus

**After=**network.target

**[Service]**

**ExecStart=/usr/bin/python3 /opt/pfSense\_exporter/pfSense\_exporter.py**

**Restart=always**

**User=ubuntu**

**WorkingDirectory=/opt/pfSense\_exporter**

**[Install]**

**WantedBy=multi-user.target**

**Activation et démarrage du service :**

**cd : sudo systemctl daemon-reload**

**sudo systemctl enable --now pfSense\_exporter**

**sudo systemctl status pfSense\_exporter**

**ECHEC comme avec le premier dépôt linux.**

## On purge proprement :

Supprimer le service systemd :

```
cd : sudo systemctl stop pfsense_exporter  
      sudo systemctl disable pfsense_exporter  
      sudo rm /etc/systemd/system/pfsense_exporter.service  
      sudo systemctl daemon-reload
```

Supprimer les fichiers Python:

```
cd : sudo rm -rf /opt/pfsense_exporter/
```

Supprimer les logs:

```
cd : sudo journalctl --vacuum-time=1s
```

Configuration SNMP côté pfSense : **pourquoi utiliser SNMP ?**

### Usage

Exploiter le script `pfsense_exporter.py` avec les métriques `pf_` (comme dans ton JSON Grafana)

Récupérer via SNMP des métriques système ou bas-niveau (uptime, interfaces, CPU, etc.)

### Action

PAS besoin de Telegraf

Telegraf est utile

### Intérêt :

#### Donnée à récupérer

États de firewall pfSense

Statut interfaces SNMP

UpTime général

CPU, RAM (pfSense)

#### Méthode

API (HTTP)

SNMP (port 161)

SNMP

SNMP

#### Outil

`pfsense_exporter.py`

Telegraf

Telegraf

Telegraf

### Prérequis :

Installer SNMP :

```
cd : sudo apt install snmp -y
```

## 1 Sur pfsense : Activer SNMP

- Menu : Services > SNMP
- Activez le service
- Port : 161, community : public (ou sécurisé)
- Activez les modules d'interface et de système

## Mise à jour - Tests SNMP après ouverture du port 161/UDP

The screenshot shows the pfSense web interface under the 'Manage Security Group Rules' section for the 'visio-monitoring-group'. The table displays the following rules:

Index	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
1	Egress	IPv4	Any	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
2	Egress	IPv6	Any	Any	::/0	-	-	<button>Delete Rule</button>
3	Ingress	IPv4	TCP	3000	0.0.0.0/0	-	Grafana	<button>Delete Rule</button>
4	Ingress	IPv4	TCP	4443	0.0.0.0/0	-	Custom HTTPS / WebSocket et reverse proxy	<button>Delete Rule</button>
5	Ingress	IPv4	UDP	161	0.0.0.0/0	-	allow-snmp	<button>Delete Rule</button>
6	Ingress	IPv4	UDP	10000	0.0.0.0/0	-	Jitsi meet	<button>Delete Rule</button>

## Ouverture du port 161

Une règle Ingress a bien été ajoutée dans le groupe de sécurité visio-monitoring-group pour autoriser le trafic entrant sur le port 161/UDP depuis toutes les IPs (CIDR 0.0.0.0/0).

- Tests snmpwalk Trois commandes ont été lancées successivement :
- `snmpwalk -v2c -c prom-readonly 37.156.46.238 1.3.6.1.2.1.1`
- `snmpwalk -v2c -c prom-readonly 192.168.1.1 1.3.6.1.2.1.1`
- `snmpwalk -v2c -c prom-readonly 192.168.0.1 1.3.6.1.2.1.1`

Toutes ont retourné la même erreur :

**Timeout: No Response from [IP ciblée]**

- Hypothèses actuelles sur l'échec du test
  - L'interface pfSense à l'adresse supposée 192.168.1.1 ou 192.168.0.1 est injoignable depuis la VM Ubuntu car elle n'est pas sur le même sous-réseau virtuel.

- Le service SNMP n'est pas activé ou autorisé sur pfSense.
  - L'adresse publique 37.156.46.238 ne correspond pas directement à pfSense mais à une VM d'application (Jitsi).
  - Le filtrage résiduel par un autre pare-feu ou une couche d'abstraction empêche la communication SNMP.
- Prochaine étape
    - Vérification de l'adresse réelle de pfSense (via console, routage, ou listing Infomaniak s'il y a accès).
    - Si accès impossible : documenter ce blocage comme limite d'infrastructure dans un environnement managé.
    - Continuer à utiliser uniquement Telegraf, Node Exporter et Blackbox Exporter sur l'instance Ubuntu pour le monitoring.
  - Conclusion temporaire Le monitoring SNMP depuis la VM vers pfSense est non fonctionnel à ce stade, malgré les ouvertures de port. Une simulation sera envisagée pour compléter le rapport de soutenance.

## 2 Sur le serveur Ubuntu :

Contrôle de l'existence de telegraf :

**cd** : telegraf –version

**réponse attendue** : Telegraf 1.28.2 (git: HEAD 3d7aa20c)

Installer Telegraf :

**cd** : sudo apt update

sudo apt install -y telegraf

**réponse** : le dépôt n'est pas connu d'apt == E: Unable to locate package telegraf

Étapes pour installer Telegraf sur Ubuntu :

**# 1. Importer la clé GPG d'InfluxData**

**cd** : wget -qO- https://repos.influxdata.com/influxdata-archive\_compatible.key | sudo gpg --dearmor -o /etc/apt/keyrings/influxdata.gpg **ok**

**# 2. Ajouter le dépôt InfluxData pour Ubuntu (remplacer \$(lsb\_release -cs) si besoin)**

**cd** : echo "deb [signed-by=/etc/apt/keyrings/influxdata.gpg]

https://repos.influxdata.com/ubuntu \$(lsb\_release -cs) stable" | sudo tee

/etc/apt/sources.list.d/influxdata.list **ok**

```
Setting up telegraf (1.35.2-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/telegraf.service → /usr/lib/systemd/system/telegraf.service.
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@jitsi-tercius:~$
```

### # 3. Mettre à jour les paquets

cd : sudo apt update ok

```
ubuntu@jitsi-tercium:~$ sudo apt update
Hit:1 https://apt.grafana.com stable InRelease
Hit:2 https://repos.influxdata.com/ubuntu focal InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 http://az-1.clouds.archive.ubuntu.com/ubuntu noble InRelease
Hit:6 http://az-1.clouds.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:7 http://az-1.clouds.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb InRelease
Hit:8 https://download.jitsi.org stable/ InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
74 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: https://pkgs.k8s.io/core:/stable:/v1.30/deb/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg),
see the DEPRECATION section in apt-key(8) for details.
```

### # 4. Installer Telegraf

cd : sudo apt install telegraf -y ok

```
ubuntu@jitsi-tercium:~$ sudo apt install telegraf -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
telegraf is already the newest version (1.35.2-1).
0 upgraded, 0 newly installed, 0 to remove and 74 not upgraded.
ubuntu@jitsi-tercium:~$
```

### # 5. Vérifie le statut de Telegraf

cd : sudo systemctl status telegraf ok

```
ubuntu@jitsi-tercium:~$ sudo systemctl status telegraf
● telegraf.service - Telegraf
    Loaded: loaded (/usr/lib/systemd/system/telegraf.service; enabled; preset: enabled)
    Active: inactive (dead)
      Docs: https://github.com/influxdata/telegraf
ubuntu@jitsi-tercium:~$
```

### # 6. Activer et démarrer Telegraf

cd : sudo systemctl enable --now telegraf

```
ubuntu@jitsi-tercium:~$ sudo systemctl status telegraf.service
● telegraf.service - Telegraf
    Loaded: loaded (/usr/lib/systemd/system/telegraf.service; enabled; preset: enabled)
    Active: failed (Result: exit-code) since Fri 2025-07-11 14:34:17 UTC; 33s ago
      Docs: https://github.com/influxdata/telegraf
   Process: 79986 ExecStart=/usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-directory /etc/telegraf/telegraf.➤
 Main PID: 79986 (code=exited, status=1/FAILURE)
        CPU: 110ms

Jul 11 14:34:17 jitsi-tercium systemd[1]: telegraf.service: Scheduled restart job, restart counter is at 5.
Jul 11 14:34:17 jitsi-tercium systemd[1]: telegraf.service: Start request repeated too quickly.
Jul 11 14:34:17 jitsi-tercium systemd[1]: telegraf.service: Failed with result 'exit-code'.
Jul 11 14:34:17 jitsi-tercium systemd[1]: Failed to start telegraf.service - Telegraf.
```

échec : Afficher le statut du service avec erreur détaillée :

cd : sudo systemctl status telegraf.service

**Exposer Telegraf à Prometheus :**

**Créer le dossier prometheus :**

**cd :** sudo mkdir -p /etc/prometheus

**Configuration de base de Prometheus :**

**yaml :**

**global:**

**scrape\_interval: 15s**

**evaluation\_interval: 15s**

**scrape\_configs:**

**- job\_name: 'prometheus'**

**static\_configs:**

**- targets: ['localhost:9090']**

**Étapes pour l'ajouter avec nano :**

**cd :** sudo nano /etc/prometheus/prometheus.yml

**Redémarrer Prometheus:**

**cd :** sudo systemctl restart prometheus



### Installation de IPATABLES : IMPOSSIBLE Actuellement.

Installation de l'exporter : Il y une obsolescence du dépôt ainsi qu'une demande de code absurde sur du github.

PRÉREQUIS :

```
cd : sudo apt update  
      sudo apt install -y golang git make
```

VERIFICATION :

```
cd : git --version  
      go version  
      make --version
```

NETTOYAGE :

```
cd :  
      sudo rm -rf iptables_exporter
```

COLONAGE & COMPILEMENT :

```
cd :  
      cd iptables_exporter  
      make
```

Utilisateur dédié :

```
cd : sudo useradd -rs /bin/false iptables_exporter  
      sudo chown -R iptables_exporter:iptables_exporter /opt/iptables_exporter
```

Service :

```
cd : sudo nano /etc/systemd/system/iptables_exporter.service
```

contenu :

```
[Unit]  
Description=Iptables Exporter  
After=network.target
```

[Service]

```
User=iptables_exporter  
ExecStart=/opt/iptables_exporter/iptables_exporter
```

[Install]

```
WantedBy=multi-user.target
```

```
sudo systemctl daemon-reload  
sudo systemctl enable --now iptables_exporter
```

Autoriser le port 9145 choisi dans Service Group :

```
cd : sudo ufw allow 9145/tcp
```

Vérification du lancement du service :

```
cd : sudo systemctl status iptables_exporter
```



Jour-9 Tercium 2<sup>nde</sup> semaine | monitoring

## Connexion SSH en Git Bash :

```
cd : ssh -i "/c/Users/test/Documents/Tercium_Stage/ssh_keys/tercium-instance_key"
ubuntu@37.156.46.238:
```

### **Connexion SSH en Powershell :**

```
cd : ssh -i "C:\Users\test\Documents\Tercium_Stage\ssh_keys\tercium-instance_key"
ubuntu@37.156.46.238
```

## \*Contrôle des ports : Permissions

<b>sudo ufw allow 9091/tcp</b>	<b>Prometheus</b>
<b>sudo ufw allow 9100/tcp</b>	<b>Node Exporter</b>
<b>sudo ufw allow 9115/tcp</b>	<b>Blackbox Exporter</b>
<b>sudo ufw allow 3000/tcp</b>	<b>Grafana</b>
<b>sudo ufw allow 9145/tcp</b>	<b>Iptables_Exporter</b>

```
cd : sudo ufw allow 9091/tcp  
      sudo ufw allow 9100/tcp  
      sudo ufw allow 9115/tcp  
      sudo ufw allow 3000/tcp  
      sudo ufw allow 9145/tcp / ECHEC
```

```
ubuntu@jitsi-tercius:~$ sudo ufw allow 9091/tcp
sudo ufw allow 9100/tcp
sudo ufw allow 9115/tcp
sudo ufw allow 3000/tcp
sudo ufw allow 9145/tcp
Skipping adding existing rule
Skipping adding existing rule (v6)
Rules updated
Rules updated (v6)
ubuntu@jitsi-tercius:~$
```

\*Contrôle des ports :Listen

**cd : sudo ss -tulnp | grep -E '9001|9100|9115|3000|9145'**

```
ubuntu@jitsi-tercium:~$ # Vérifie l'écoute
sudo ss -tulnp | grep 9091

# Vérifie la réponse HTTP
curl http://localhost:9091

# Vérifie l'accessibilité depuis l'extérieur
curl http://[IP_PUBLIQUE]:9091
tcp        LISTEN      0      4096          *:9091              *:*      users:(("prometheus",pid=258337
,fd=7))
<a href="/graph">Found</a>.

curl: (3) bad range in URL position 9:
http://[IP_PUBLIQUE]:9091
^
ubuntu@jitsi-tercium:~$
```

```
ubuntu@jitsi-tercium:~$ sudo ss -tulnp | grep -E '9091|9100|9115|3000|9145'
tcp    LISTEN  0      4096                           *:9091          *:*      users:(("prometheus",pid=258337
,fd=7))
tcp    LISTEN  0      4096                           *:9100          *:*      users:(("node_exporter",pid=752
16,fd=3))
```

```
server.service.
ubuntu@jitsi-tercium:~$ sudo ss -tulnp | grep 3000
tcp    LISTEN  0      4096                           *:3000          *:*      users:(("grafana",pid=265585,fd
=13))
```

**Prometheus :** Après modifications contenant port et les logs de blackbox\_http et node\_exporter l'on peut relancer et tester l'interface.

**Redémarrer :**

**cd :** sudo systemctl restart prometheus

**Vérifier l'état :**

**cd :** sudo systemctl status prometheus  
curl <http://localhost:9091/targets>

**Navigateur web :**

**cd :** <http://37.156.46.238:9091/targets>

## Sécurisation HTTPS de Prometheus et Grafana

### A. Reverse proxy avec NGINX pour Grafana (port 3000)

Grafana a besoin d'alimenter ses analyses via des dashboard au standard Json. Ceux-ci doivent être personnalisables et interactifs. Les buts premiers sont la surveillance et l'analyse et surtout une capacité de corriger en temps réel des problèmes et l'expérience aidant être proactif.

## Usages principaux des dashboards dans Grafana

### 1. Supervision temps réel (real-time monitoring)

- Affichage de métriques en continu provenant de sources comme Prometheus, InfluxDB, Loki, Elasticsearch, etc.
- Exemple : suivi de l'utilisation CPU/RAM/disque d'un serveur.

### 2. Détection d'anomalies

- Mise en évidence de comportements anormaux via des seuils, des pics, ou des écarts.
- Exemples :
  - Un nombre élevé de requêtes 500 (erreurs HTTP).
  - Une latence anormalement élevée sur un service.

### **3. Alerting visuel + automatisé**

- **Couplage avec des alertes configurées (via Alertmanager, mail, Slack, etc.).**
- **Les dashboards permettent de visualiser les alertes déclenchées et leurs causes.**

### **4. Diagnostic et analyse post-incident (forensic/debug)**

- **Permet de rejouer un incident dans le temps, via le sélecteur temporel.**
- **Filtrage par label, par hôte, ou par type d'erreur.**

### **5. Rapports et communication**

- **Affichage synthétique pour les managers, RSSI, ou clients (en lecture seule).**
- **Possibilité d'export PDF, share URL, ou d'intégration dans un site via iframe.**

### **6. Vue consolidée multi-sources**

- **Intégration de plusieurs sources de données dans un même dashboard (ex. : Prometheus + Loki + Blackbox).**
- **Corrélation des événements (ex. : logs + métriques système).**

### **7. Pilotage des performances**

- **Suivi des indicateurs clés de performance (KPI) métiers ou techniques.**
- **Exemple : taux de disponibilité (uptime), temps de réponse moyen, nombre d'utilisateurs actifs.**

### **8. Suivi de conformité et d'audit**

- **Vérification de l'état de sécurité et de la conformité des systèmes (ISO 27001, RGPD, NIS2...).**
- **Exemple : intégration avec Wazuh, Suricata, ou pfSense Exporter.**

### **9. Visualisation de scénarios métier**

- **Dashboards orientés usage spécifique (ex. : flux réseau Jitsi, montée en charge d'une API, monitoring d'un cluster Kubernetes...).**
- **Utilisés pour tester et valider des stratégies de montée en charge ou de résilience.**

### **10. Maintenance préventive**

- **Identification des machines ou services qui risquent de tomber en panne (RAM saturée, partition disque à 95 %, etc.).**
- **Utilisation dans les plans de maintenance évolutive (ITIL / ISO 22301).**

#### **Exemples de widgets / panels dans un dashboard :**

- **Graphes temporels (CPU, charge, requêtes)**
- **Barres ou jauge (utilisation mémoire)**
- **Statistiques tabulaires (état des services)**
- **Logs filtrables (via Loki)**
- **Heatmaps, alert lists, pie charts, singlestats, etc.**

## Infomaniak ; la gestion des groupes de sécurité pour les ports :

Voici le **tableau récapitulatif des ports essentiels** à ouvrir par service pour ta plateforme Jitsi + Monitoring (Prometheus, Grafana, Blackbox, pfSense Exporter).

### ✓ Tableau des ports par service

Service	Port	Protocole	Type	Description
SSH (admin)	22	TCP	Ingress	Connexion au serveur
HTTP	80	TCP	Ingress	Web non chiffré (utile pour redirection)
HTTPS	443	TCP	Ingress	Accès web sécurisé (ex. visio.workeezconnect)
Grafana	3000	TCP	Ingress	Interface Web Grafana
Jitsi Media	10000	UDP	Ingress	Canal de flux audio/vidéo UDP (WebRTC)
Prometheus UI	9090 (ou 9091)	TCP	Ingress	Interface Prometheus
Node Exporter	9100	TCP	Ingress	Export métriques serveur Linux
Blackbox Exporter	9115	TCP	Ingress	Tests ICMP/HTTP externes
iptables Exporter	9145	TCP	Ingress	Export règles netfilter/iptables

### 📦 Groupes créés :

Nom du groupe	Ports inclus	Utilisation
default (désactivable)	Aucun si retiré	Peut être exclu
monitoring-group	9091, 9100, 9115, 9145	Prometheus + Exporters
monitoring-security-group	22, 80, 443	Accès standard SSH + HTTP/HTTPS
visio-monitoring-group	3000, 10000	Grafana + Jitsi UDP (media)

Project / Network / Security Groups / Manage Security Group Rule...

Manage Security Group Rules: monitoring-group (e7742305-590e-4791-b1cd-f013a0758aca)

Action	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
<input type="checkbox"/>	Egress	IPv4	Any	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
<input type="checkbox"/>	Egress	IPv6	Any	Any	::/0	-	-	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv4	TCP	9091	0.0.0.0/0	-	Prometheus	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv4	TCP	9100	0.0.0.0/0	-	Node_Export	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv4	TCP	9115	0.0.0.0/0	-	Blackbox	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv4	TCP	9145	0.0.0.0/0	-	iptables Exporter	<button>Delete Rule</button>

Ce Security Group intègre l'ensemble des ports dédiés aux outils de Monitoring.

Project / Network / Security Groups / Manage Security Group Rule...

Manage Security Group Rules: monitoring-security-group (bdd37ffd-f073-42fa-91c5-93adbc39d74b)

Action	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
<input type="checkbox"/>	Egress	IPv4	Any	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
<input type="checkbox"/>	Egress	IPv6	Any	Any	::/0	-	-	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	TCP 22 (SSH), 80, 443	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0	-	http	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv4	TCP	443 (HTTPS)	0.0.0.0/0	-	https	<button>Delete Rule</button>

Ce Security Group intègre l'ensemble des ports HTTP / HTTPS / SSH.



## Attention

- Le port **10000/UDP** est critique pour que la visioconférence fonctionne.
- **IPv6** peut être ignoré sauf cas spécifique (pas utile pour WebRTC sauf config avancée).
- L'on peut désormais supprimer "**default**" une fois **tous les groupes nécessaires rattachés**.

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
Egress	IPv6	Any	Any	::/0	-	-	<button>Delete Rule</button>
Ingress	IPv4	TCP	3000	0.0.0.0/0	-	Grafana	<button>Delete Rule</button>
Ingress	IPv4	UDP	10000	0.0.0.0/0	-	Jitsi meet	<button>Delete Rule</button>

Ce dernier Security Group intègre les ports pour Grafana & Jitsi meet.

## Automatiser avec Terraform pour instancier Infomaniak : Possible

Installer Terraform puis :

### 1/ Contenu du projet

Fichier	Rôle
<code>clouds.yaml</code>	Configuration OpenStack pour Infomaniak
<code>main.tf</code>	Création de l'instance jitsi-tercium avec le bon security_group
<code>security.tf</code>	Règles de sécurité pour Grafana (TCP 3000) et Jitsi (UDP 10000)
<code>variables.tf</code>	Variable pour le nom de la clé SSH
<code>.gitignore</code>	Ignore les fichiers .tfstate, .terraform/ et credentials sensibles

### 2/ Les étapes :

1. Configurer `clouds.yaml` localement dans : `~/.config/openstack/`.
  2. Renseigner les vraies valeurs dans `clouds.yaml` :
    - o `username`, `password`, `project_id`, etc.
  3. Initialiser le projet avec ces commandes :
- `cd` : `terraform init`  
`terraform plan`  
`terraform apply`

### Structure recommandée du projet

```
project/
  └── main.tf
  └── variables.tf
  └── terraform.tfvars.example
  └── terraform.sh
```

### 3/ Exemple de fichier : Nécessaires à Infomaniak

#### Terraform.tfvars

```
# Clé SSH utilisée pour accéder à l'instance
keypair_name = "tercium-instance_key"

# Nom de l'image (ID ou nom exact d'Infomaniak OpenStack)
image_name = "Debian 12 Bookworm"

# Nom du flavor (ex: gp1.large, cp1.medium, etc.)
flavor_name = "gp1.large"

# Nom du réseau privé ou réseau par défaut d'Infomaniak
network_name = "Ext-Net1"

# Groupe de sécurité à appliquer
security_group_name = "jitsi-monitoring-secgroup"

# Nom d'instance (machine virtuelle)
instance_name = "jitsi-tercium"
```

#### Modèles de fichier :

##### 1/ main.tf

```
provider "openstack" {
  auth_url  = var.auth_url
  tenant_name = var.project_name
  user_name  = var.username
  password   = var.password
  region     = var.region
  domain_name = var.domain_name
}

# Groupe de sécurité personnalisé
resource "openstack_networking_secgroup_v2" "jitsi_secgroup" {
  name      = "jitsi_monitoring_secgroup"
  description = "Allow SSH, HTTP, HTTPS, Prometheus, Grafana, Jitsi, Node Exporter, Blackbox"
}

# Règles de sécurité essentielles
resource "openstack_networking_secgroup_rule_v2" "rules" {
  count = length(var.security_rules)

  direction      = var.security_rules[count.index]["direction"]
  ethertype      = var.security_rules[count.index]["ethertype"]
  protocol       = var.security_rules[count.index]["protocol"]
  port_range_min = var.security_rules[count.index]["port_range_min"]
```

```

port_range_max = var.security_rules[count.index]["port_range_max"]
remote_ip_prefix = var.security_rules[count.index]["remote_ip_prefix"]
security_group_id = openstack_networking_secgroup_v2.jitsi_secgroup.id
}

# Clé SSH
resource "openstack_compute_keypair_v2" "default" {
  name      = "tercium-key"
  public_key = file(var.public_key_path)
}

# Instance Jitsi
resource "openstack_compute_instance_v2" "jitsi_vm" {
  name        = "jitsi-tercium"
  image_name   = var.image_name
  flavor_name   = var.flavor_name
  key_pair     = openstack_compute_keypair_v2.default.name
  security_groups = [openstack_networking_secgroup_v2.jitsi_secgroup.name]
  network {
    name = var.network_name
  }
}

2/ variables.tf
variable "auth_url" {}
variable "username" {}
variable "password" {}
variable "domain_name" {}
variable "project_name" {}
variable "region" {}
variable "public_key_path" {}
variable "image_name" {}
variable "flavor_name" {}
variable "network_name" {}

variable "security_rules" {
  type = list(object({
    direction      = string
    ethertype      = string
    protocol       = string
    port_range_min = number
    port_range_max = number
    remote_ip_prefix = string
  }))
}

3/ terraform.tfvars
auth_url      = "https://api.infomaniak.com/identity/v3"
username      = "TON_IDENTIFIANT"
password      = "TON_MOT_DE_PASSE"

```

```

domain_name    = "default"
project_name   = "NomDuProjet"
region        = "dc3-a"
public_key_path = "~/.ssh/id_rsa.pub"
image_name     = "Ubuntu 22.04"
flavor_name    = "a1-ram2-disk20-perf1"
network_name   = "Ext-Net"

security_rules = [
{
  direction      = "ingress"
  ethertype     = "IPv4"
  protocol      = "tcp"
  port_range_min = 22
  port_range_max = 22
  remote_ip_prefix = "0.0.0.0/0"
},
{
  direction      = "ingress"
  ethertype     = "IPv4"
  protocol      = "tcp"
  port_range_min = 80
  port_range_max = 80
  remote_ip_prefix = "0.0.0.0/0"
},
{
  direction      = "ingress"
  ethertype     = "IPv4"
  protocol      = "tcp"
  port_range_min = 443
  port_range_max = 443
  remote_ip_prefix = "0.0.0.0/0"
},
{
  direction      = "ingress"
  ethertype     = "IPv4"
  protocol      = "tcp"
  port_range_min = 3000
  port_range_max = 3000
  remote_ip_prefix = "0.0.0.0/0"
},
{
  direction      = "ingress"
  ethertype     = "IPv4"
  protocol      = "tcp"
  port_range_min = 10000
  port_range_max = 10000
  remote_ip_prefix = "0.0.0.0/0"
}
]

```

## Jour-10 Tercium 2<sup>nde</sup> semaine | monitoring routines de démarrage

Connexion SSH en Git Bash :

**cd : ssh -i "/c/Users/test/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key"**  
**ubuntu@37.156.46.238**

Connexion SSH en Powershell :

**cd : ssh -i "C:\Users\test\Documents\Tercium\_Stage\ssh\_keys\tercium-instance\_key"**  
**ubuntu@37.156.46.238**

Relancer les services système via le terminal :

**cd : sudo systemctl restart prometheus**  
**sudo systemctl restart node\_exporter**  
**sudo systemctl restart blackbox\_exporter**

Pour vérifier qu'ils tournent :

**cd : sudo systemctl status prometheus**  
**sudo systemctl status node\_exporter**  
**sudo systemctl status blackbox\_exporter**

Bonnes pratiques :

- Prometheus doit être actif pour collecter les métriques.
- Node Exporter doit tourner sur chaque machine à surveiller.
- Blackbox Exporter doit être lancé une fois (souvent sur la même machine que Prometheus) pour faire les tests de ping, HTTP, TCP, etc.
- Grafana se connecte à Prometheus. Il n'a pas besoin d'être redémarré sauf en cas de changement de configuration du data source ou de plantage.

Astuce : automatisation au démarrage :

**cd : sudo systemctl enable prometheus**  
**sudo systemctl enable node\_exporter**  
**sudo systemctl enable blackbox\_exporter**

Accès aux interfaces Web :

Si tous les services sont actifs, l'on peut accéder aux interfaces suivantes :

Composant	URL locale	URL distante (publique)
Prometheus	<a href="http://localhost:9090">http://localhost:9090</a>	<a href="http://37.156.46.238:9090">http://37.156.46.238:9090</a> ( <i>si exposé</i> )
Node Exporter	<a href="http://localhost:9100">http://localhost:9100</a>	<a href="http://37.156.46.238:9100">http://37.156.46.238:9100</a> ( <i>si exposé</i> )
Blackbox Exporter	<a href="http://localhost:9115">http://localhost:9115</a>	<a href="http://37.156.46.238:9115">http://37.156.46.238:9115</a> ( <i>si exposé</i> )
Grafana	<a href="http://localhost:3000">http://localhost:3000</a>	<a href="http://37.156.46.238:3000">http://37.156.46.238:3000</a> 

## TESTS :

Vérifie que ton service est bien bindé sur ce port :

Exemple avec NGINX : **Interdit ici**

**cd** : sudo netstat -tulnp | grep 4433

ou

**cd** : sudo ss -tuln | grep 4433

Test à distance depuis une autre machine :

- règle de composition nc -vz <IP> <PORT>

**cd** : nc -vz 37.156.46.238 4433

Sortie nc -vz

Connection to 37.156.46.238 port 4433  
succeeded!

Connection refused

Connection timed out

Signification technique

⚠ Le port est ouvert et une application y écoute

🚫 Aucun service n'écoute sur ce port

✗ Le port est filtré (firewall ou IP injoignable)

ou via navigateur :

<https://37.156.46.238:4433>

Manuel d'usage de visio.workeezconnect.fr

Récupération et analyse des données monitoring :

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9115/probe module="http_2xx" target="https://example.com"	UP	instance="https://example.com" job="Blackbox http"	1.482s ago	480.962ms	
http://localhost:9192/metrics	DOWN	instance="localhost:9192" job="pfSense_exporter"	4.979s ago	0.826ms	Get "http://localhost:9192/metrics": dial tcp [:1]:9192: connect: connection refused
http://localhost:9091/metrics	UP	instance="localhost:9091" job="prometheus"	9.612s ago	4.222ms	

**Problème pfsense\_exporter est inactif : Pourquoi ces commandes ne fonctionnent pas ?**

**cd : sudo systemctl status pfsense\_exporter  
sudo systemctl restart pfsense\_exporter**

**Retour ce service de monitoring du 8 ième jour : Je complète et je modifie.**

# Jour-11 Tercium 3ième semaine | fin installation monitoring

Voir plus haut pour les relances qu'il faudra totalement automatiser.

Résumé des fonctions logs via les services avant relance d'installation de telegraf :

Telegraf, comparer à **Node Exporter**, **Blackbox Exporter**, et à d'autres (type **Filebeat**, **Logstash**, etc.).

## 1. Rôle de Telegraf

Telegraf est un **agent léger de collecte de métriques et de logs**. Il fait partie de la suite TICK (Telegraf, InfluxDB, Chronograf, Kapacitor), mais peut fonctionner **indépendamment** et exporter vers Prometheus ou Grafana.

**Il collecte, transforme et envoie des données.**

- **Collecte** → via des *plugins d'entrée* (inputs)
- **Traitement / mise en forme** → *plugins de transformation*
- **Envoi** → via des *plugins de sortie* (outputs)

## 2. Comparatif simplifié des outils de monitoring

Outil	Fonction principale	Type de données	Spécificité
Telegraf	Agent polyvalent de collecte	Métriques système, réseau, SNMP, Docker, logs, etc.	Très personnalisable (plugins)
Node Exporter	Exporter Prometheus pour Linux	Métriques système Linux	Simplicité, usage standard Prometheus
Blackbox Exporter	Vérification de l'accessibilité (ping, HTTP, TCP)	Résultats de tests "extérieurs" (uptime)	Surveillance type "boîte noire"
Filebeat	Collecte de fichiers de log (ELK/EFK)	Logs (texte brut, JSON, syslog, nginx...)	Léger, rapide, dédié aux logs
Logstash	Pipeline de traitement de logs complexe	Logs enrichis	Peut transformer, parser, router
Fluentd	Agent/log router JSON & cloud-friendly	Logs/streams structurés	Flexible et orienté cloud

### 3. Telegraf : Architecture des plugins

- ◆ Exemples de *plugins d'entrée* :

```
[[inputs.cpu]]  
  percpu = true  
  totalcpu = true  
  
[[inputs.mem]]  
[[inputs.net]]  
[[inputs.disk]]  
[[inputs.system]]  
[[inputs.snmp]]  
  agents = [ "192.168.1.1:161" ]  
  version = 2
```

- ◆ Plugin de sortie (*Prometheus ou InfluxDB*) :

```
[[outputs.prometheus_client]]  
  listen = ":9273"  
  
# ou :  
[[outputs.influxdb]]  
  urls = ["http://localhost:8086"]
```

## Telegraf peut aussi lire des logs si nécessaire

Exemple :

```
[[inputs.tail]]  
  files = ["/var/log/syslog"]  
  from_beginning = false  
  name_override = "syslog_logs"
```

## En résumé

Agent	Pour Prometheus ?	Pour logs ?	Configuration
Node Exporter	Oui	Non	Aucune/modeste
Telegraf	Oui/Non (pas exclusivement)	Oui	Config très riche via fichier .conf
Blackbox	Oui	Non	prometheus.yml uniquement

 Solution directe et stable de Telegraf :

1. Sauvegarder la configuration existante :

cd : sudo mv /etc/telegraf/telegraf.conf /etc/telegraf/telegraf.conf.bak

2. Créer un dossier de configuration Telegraf :

cd : sudo mkdir -p /etc/telegraf

Configurer Telegraf :

cd : sudo nano /etc/telegraf/telegraf.conf

```
[agent]
interval = "10s"
round_interval = true
metric_batch_size = 1000
metric_buffer_limit = 10000
collection_jitter = "0s"
flush_interval = "10s"
flush_jitter = "0s"
precision = ""
hostname = ""
omit_hostname = false

[[outputs.prometheus_client]]
listen = ":9273"
path = "/metrics"
expiration_interval = "60s"
collectors_exclude = ["gocollector", "process"]

[[inputs.cpu]]
percpu = true
totalcpu = true
collect_cpu_time = false
report_active = false

[[inputs.mem]]
[[inputs.disk]]
ignore_fs = ["tmpfs", "devtmpfs", "overlay"]

[[inputs.net]]
[[inputs.system]]
```

Redémarrer Telegraf & vérifier son status:

cd : sudo systemctl restart telegraf  
sudo systemctl status telegraf

Tester l'export local :

cd : curl http://localhost:9273/metrics

Vérifions le fichier de configuration de Prometheus, pas son répertoire mais le fichier:  
cd: sudo nano /etc/prometheus/prometheus.yml

```
GNU nano 7.2          /etc/prometheus/prometheus.yml
- targets: ['localhost:9090']

# 2. Node Exporter (CPU, RAM, disque, etc.)
- job_name: 'node_exporter'
  static_configs:
    - targets: ['localhost:9100']

# 3. Blackbox Exporter (surveillance de services distants)
- job_name: 'blackbox_http'
  metrics_path: /probe
  params:
    module: [http_2xx]
  static_configs:
    - targets:
        - http://localhost      # à adapter
        - https://visio.workeezconnect.fr
  relabel_configs:
    - source_labels: [__address__]
      target_label: __param_target
    - source_labels: [__param_target]
      target_label: instance
    - target_label: __address__
      replacement: localhost:9115 # Adresse du blackbox exporter

# 4. (futur) Iptables Exporter – si compilé
- job_name: 'iptables_exporter'
  static_configs:
    - targets: ['localhost:9145']

# 5. Telegraf
- job_name: 'telegraf'
  static_configs:
    - targets: ['localhost:9273']
```

Puis installer le dashboard via l'import de Grafana et tester:

Vérifier que Telegraf écoute bien sur un port Prometheus-compatible :  
cd : sudo lsof -i -P -n | grep telegraf

```
ubuntu@jitsi-tercium:~$ sudo lsof -i -P -n | grep telegraf
telegraf  25533      telegraf    6u  IPv6 392825      0t0  TCP *:9273 (LISTEN)
```

- Objectif : lister les fichiers ouverts par des processus (lsof) en filtrant ceux liés à des connexions réseau (-i) sur des ports (avec ports affichés en clair grâce à -P) sans résolution DNS (-n).
- Usage : Diagnostique fin, montre quel *processus exact* est à l'écoute d'un port (ici telegraf avec son PID 25533 sur TCP \*:9273).
- Avantage : on voit que c'est bien le binaire telegraf qui a ouvert le port.

Vérifier que Prometheus Scrape bien Telegraf via m'IP publique :

cd : <http://37.156.46.238:9091/targets> : NON

Il doit avoir cet ajout dans son code :du fichier : **prometheus.yml**

- job\_name: 'telegraf'  
static\_configs:  
- targets: ['localhost:9273'] Puis redémarrer

Il faut ajouter une règle au Security Group :

The screenshot shows a network configuration interface with a sidebar on the left containing categories like Project, API Access, Compute, Volumes, Network, and Security Groups. The Security Groups section is highlighted. In the main area, it says "Manage Security Group Rules: monitoring-group (e7742305-590e-4791-b1cd-f013a0758aca)". Below this is a table titled "Displaying 8 items" with columns: Direction, Ether Type, IP Protocol, Port Range, Remote IP Prefix, Remote Security Group, and Description. The table lists eight rules:

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description
Egress	IPv4	Any	Any	0.0.0.0/0	-	-
Egress	IPv6	Any	Any	::/0	-	-
Ingress	IPv4	TCP	9091	0.0.0.0/0	-	Prometheus
Ingress	IPv4	TCP	9100	0.0.0.0/0	-	Node_Exporte
Ingress	IPv4	TCP	9115	0.0.0.0/0	-	Blackbox
Ingress	IPv4	TCP	9145	0.0.0.0/0	-	iptables Exporter
Ingress	IPv4	TCP	9192	0.0.0.0/0	-	allow-pfsense-exporter
Ingress	IPv4	TCP	9273	0.0.0.0/0	-	Telegraf

Redémarrer Telegraf :

cd : sudo systemctl restart telegraf  
sudo systemctl status telegraf

Vérifier que le port 9273 est exposé :

cd : ss -tuln | grep 9273

```
ubuntu@jitsi-tercium:~$ sudo systemctl restart telegraf
ubuntu@jitsi-tercium:~$ ss -tuln | grep 9273
tcp    LISTEN  0      4096                           *:9273
```

## Nuance à comprendre :

Commande	Vérifie port ouvert ?	Affiche PID / processus ?	Utilité
`lsof -i -P -n	grep telegraf	<input checked="" type="checkbox"/> Oui	<input checked="" type="checkbox"/> Oui
`ss -tuln	grep 9273`	<input checked="" type="checkbox"/> Oui	<input checked="" type="checkbox"/> Non

Le -p permet d'afficher aussi le nom et le PID du processus (comme lsof).

Test Grafana ave une session visio.workeez.connect.fr entre deux PC.

blackbox_http (1/1 up)					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9115/probe	UP	instance="https://example.com" job="blackbox_http" target="https://example.com"	35.553s ago	554.567ms	

pfsense_exporter (0/1 up)					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9192/metrics	DOWN	instance="localhost:9192" job="pfsense_exporter"	24.45s ago	0.794ms	Get "http://localhost:9192/metrics": dial tcp [::1]:9192: connect: connection refused

prometheus (1/1 up)					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9091/metrics	UP	instance="localhost:9091" job="prometheus"	28.677s ago	5.147ms	

telegraf (1/1 up)					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9273/metrics	UP	instance="localhost:9273" job="telegraf"	25.237s ago	3.303ms	

Pour pouvoir utiliser pfsense indirectement il faut rajouter à Telegraf à la fin cette extension. Attention besoin possible de MIBs !

```
toml

[[inputs.snmp]]
agents = [ "IP_PFSENSE:161" ]
version = 2
community = "public" # ou ta chaîne SNMP
name = "pfsense"

[[inputs.snmp.field]]
name = "uptime"
oid = "1.3.6.1.2.1.1.3.0"
is_tag = true

[[inputs.snmp.table]]
name = "interface"
oid = "1.3.6.1.2.1.2.2"
```

Puis relance Telegraf :

L'option pfSense peut revenir en incluant non pas l'installation via github dépôt inopérant ni une version python elle aussi inerte mais via des inputs dans [telegraf.conf](#).

Bloc [inputs.snmp] — interrogation de pfSense via SNMP :

```
[[inputs.snmp]]
agents = [ "192.168.1.1:161" ]          # ← Remplace par l'IP réelle de pfSense
version = 2
community = "prom-readonly"
name = "pfSense"

[[inputs.snmp.field]]
name = "uptime"
oid = "1.3.6.1.2.1.1.3.0"           # sysUpTimeInstance

[[inputs.snmp.field]]
name = "hostname"
oid = "1.3.6.1.2.1.1.5.0"           # sysName

[[inputs.snmp.table]]
name = "interface"
inherit_tags = [ "hostname" ]
oid = "1.3.6.1.2.1.2.2"             # ifTable

[[inputs.snmp.table.field]]
name = "ifDescr"
oid = "1.3.6.1.2.1.2.2.1.2"

[[inputs.snmp.table.field]]
name = "ifInOctets"
oid = "1.3.6.1.2.1.2.2.1.10"

[[inputs.snmp.table.field]]
name = "ifOutOctets"
oid = "1.3.6.1.2.1.2.2.1.16"

[[outputs.prometheus_client]]
listen = ":9273"
path = "/metrics"
expiration_interval = "60s"
collectors_exclude = [ "gocollector", "process" ]
```

Redémarrage après modification :

**cd : sudo systemctl daemon-reexec  
sudo systemctl restart telegraf**

## Fonction de systemctl daemon-reexec

Cette commande : cd

Recharge complètement le binaire systemd (le gestionnaire d'unités et services), sans redémarrer le système.

Contexte technique :

- Quand tu modifies des fichiers liés à systemd (unités .service, chemins, permissions système, socket activation...),
- systemd peut être encore en train d'utiliser une ancienne version de lui-même en mémoire,
- daemon-reexec demande explicitement à systemd de se recharger entièrement à partir du disque, y compris les binaires et les données de configuration.

À distinguer de :

Commande

Fonction

[sudo systemctl daemon-reexec](#)

Recharge le processus systemd lui-même (rarement nécessaire, sauf modif profonde)

[sudo systemctl daemon-reload](#)

Recharge les fichiers d'unités après modification (plus courant)

[sudo systemctl restart <service>](#)

Redémarre un service spécifique

## Jour-12 Tercium 3ième semaine | installation monitoring recap

Connexion SSH en Git Bash :

**cd : ssh -i "/c/Users/test/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key"**  
**ubuntu@37.156.46.238**

Connexion SSH en Powershell :

**cd : ssh -i "C:\Users\test\Documents\Tercium\_Stage\ssh\_keys\tercium-instance\_key"**  
**ubuntu@37.156.46.238**

Commandes à automatiser via un Script :

Relancer les services système via le terminal :

**cd : sudo systemctl restart prometheus**  
**sudo systemctl restart node\_exporter**  
**sudo systemctl restart blackbox\_exporter**  
**sudo systemctl restart telegraf**

Pour vérifier qu'ils tournent :

**cd : sudo systemctl status prometheus**  
**sudo systemctl status node\_exporter**  
**sudo systemctl status blackbox\_exporter**  
**sudo systemctl status telegraf**

Bonnes pratiques :

- Prometheus doit être actif pour collecter les métriques.
- Node Exporter doit tourner sur chaque machine à surveiller.
- Blackbox Exporter doit être lancé une fois (souvent sur la même machine que Prometheus) pour faire les tests de ping, HTTP, TCP, etc.
- Grafana se connecte à Prometheus. Il n'a pas besoin d'être redémarré sauf en cas de changement de configuration du data source ou de plantage.

Astuce : automatisation au démarrage :

**cd : sudo systemctl enable prometheus**  
**sudo systemctl enable node\_exporter**  
**sudo systemctl enable blackbox\_exporter**  
**sudo systemctl enable telegraf**

## Accès aux interfaces Web :

Si tous les services sont actifs, l'on peut accéder aux interfaces suivantes :

Composant	URL locale	URL distante (publique)
Prometheus	<a href="http://localhost:9090">http://localhost:9090</a>	<a href="http://37.156.46.238:9091">http://37.156.46.238:9091</a> ( <i>si exposé</i> )
Node Exporter	<a href="http://localhost:9100">http://localhost:9100</a>	<a href="http://37.156.46.238:9100">http://37.156.46.238:9100</a> ( <i>si exposé</i> )
Blackbox Exporter	<a href="http://localhost:9115">http://localhost:9115</a>	<a href="http://37.156.46.238:9115">http://37.156.46.238:9115</a> ( <i>si exposé</i> )
Grafana	<a href="http://localhost:3000">http://localhost:3000</a>	<a href="http://37.156.46.238:3000">http://37.156.46.238:3000</a> 

Vérifier la persistance après redémarrage :

**cd** : sudo systemctl list-unit-files | grep enabled

Puis

**cd** : sudo systemctl status <service>

Journal permanent :

**cd** : journalctl -u telegraf -n 50 --no-pager

Les ports sont bien ouverts, vérifier :

**cd** : sudo ss -tuln | grep -E '9273|9091|9115|9100|3000|4443'

Si MIBs correctement chargées :

- inputs.snmp peut s'initialiser → pas d'erreur OID unknown.
- Telegraf démarre sans crash.
- Grafana retrouve automatiquement les métriques SNMP de pfSense.
- Donc :
  - pfSense s'affiche en *UP* dans le dashboard (si reachable).
  - Telegraf = service active (running).
  - Port :9273 expose bien les métriques pour Prometheus.

Si MIBs absentes ou mal résolues :

- inputs.snmp → échec initialisation (Unknown Object Identifier).
- Telegraf → crash (exit status 1).
- Donc :
  - Grafana = pfSense en rouge (aucune métrique visible).
  - Telegraf = service failed.
  - Port :9273 = fermé ou vide.

Pour réparer

1. Installe les MIBs nécessaires :

**cd** : sudo apt install snmp-mibs-downloader  
sudo download-mibs

2. Activer la lecture des noms de MIBs (désactivée par défaut dans snmp.conf) :

**cd** : sudo nano /etc/snmp/snmp.conf

# mibs : on le commente

### 3. Redémarrer Telegraf :

**cd** : sudo systemctl restart telegraf

### Récapitulatif :

Résumé Des Ports Et Services Prometheus			
□	Service	Port	Rôle
1	Prometheus	9091	Surveillance des métriques internes
2	Telegraf	9273	Métriques système via SNMP, CPU, RAM, etc.
3	Node Exporter	9100	Exportation des métriques système Linux
4	pfSense Exporter	9192	Exportation des métriques de pare-feu pfSense
5	Blackbox Exporter	9115	Tests de disponibilité HTTP(S), ping, DNS

### Explications :

```
# Scrape configuration for Prometheus Lui-même
# - job_name: 'prometheus'
#   static_configs:
#     - targets: ['localhost:9091']

# Node_exporter pour CPU, RAM, etc.
# - job_name: 'node_exporter'
#   static_configs:
#     - targets: ['IP_PUBLIQUE:9100']

# Telegraf (via SNMP ou autres plugins système)
# - job_name: 'telegraf'
#   static_configs:
#     - targets: ['IP_PUBLIQUE:9273']

# pfSense Exporter (pare-feu)
# - job_name: 'pfsense'
#   static_configs:
#     - targets: ['IP_PUBLIQUE:9192']

# Blackbox Exporter (tests réseau, HTTP, DNS)
# - job_name: 'blackbox'
#   metrics_path: /probe
#   params:
#     module: [http_2xx]
#   static_configs:
#     - targets:
#       - http://ton_domaine_ou_ip_publique
#   relabel_configs:
#     - source_labels: [__address__]
#       target_label: __param_target
#     - source_labels: [__param_target]
#       target_label: instance
#     - target_label: __address__
#       replacement: IP_PUBLIQUE:9115
```

## Résultat visible sur Prometheus :

Targets					
All scrape pools		All	Unhealthy	Collapse All	Filter by endpoint or labels
<input checked="" type="checkbox"/> Unknown <input checked="" type="checkbox"/> Unhealthy <input checked="" type="checkbox"/> Healthy					
<b>blackbox_http (1/1 up)</b> <a href="#">show less</a>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9115/probe module="http_box" target="https://example.com"	UP	instance="https://example.com" job="blackbox_http" target="https://example.com"	1.73s ago	299.000ms	
<b>pfsense_exporter (0/1 up)</b> <a href="#">show less</a>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9192/metrics	DOWN	instance="localhost:9192" job="pfsense_exporter"	4.566s ago	1.013ms	Get "http://localhost:9192/metrics": dial tcp [:1]:9192: connect: connection refused
<b>prometheus (1/1 up)</b> <a href="#">show less</a>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9091/metrics	UP	instance="localhost:9091" job="prometheus"	9.198s ago	3.969ms	
<b>telegraf (1/1 up)</b> <a href="#">show less</a>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9273/metrics	UP	instance="localhost:9273" job="telegraf"	5.757s ago	2.011ms	

Donc pfsense ne répond pas voir note en annexe.

Cependant : choix des sorties = up :

The screenshot shows the Grafana Explore interface. At the top, there's a search bar and various navigation buttons. A prominent message box says: "Explore Metrics, Logs, Traces and Profiles have moved! Looking for the Grafana Explore apps? They are now called the Grafana Drilldown apps and can be found under Menu > Drilldown. Go to Grafana Drilldown". Below this, there are two query panels labeled A and B, both for the 'prometheus' job. Panel A has a query input "up" and a dropdown "Metric: up". Panel B also has a query input "up" and a dropdown "Metric: up". Both panels include "Label filters" and "Operations" sections. At the bottom, there are buttons for "+ Add query" and "Query inspector".

## Diagramme et résultat :



Un rapport d'incident a été produit qui alimentera cette partie.

Objectif à court terme : load balancing fonctionnel demain

Je dois disposer d'un service Prometheus propre et stable pour pouvoir monitorer :

- CPU / RAM / Réseau via node\_exporter et telegraf
- Résultat de tests blackbox (HTTP/HTTPS)
- Une cible Jitsi visio active

Décision : Killprocessus / relance de l'instance via un shut-off puis start et relance et vérification des processus de monitoring :

État actuel :

- Prometheus tournait manuellement → OK.
- En mode service (systemd) → problèmes de config, de lecture, ou de port déjà pris.
- node\_exporter tourne, mais invisible.
- Tu es sur une IP publique (pas de localhost).
- Risque de confusion si plusieurs processus tournent (service actif + process manuel).

RECOMMANDATION IMMÉDIATE : stopper l'instance

Pourquoi ?

- Un port 9091 propre.
- Redémarrage net des binaires Prometheus et exporters.
- Eviter les interférences invisibles ou conflits de socket réseau.
- Plus sûr que de forcer des kill -9.

## Plan de redémarrage clair :

1. On reprend :
    - o Redémarrer l'instance manuellement ou depuis l'interface cloud. **ok**
    - o Lancer ce check minimal :
      - sudo systemctl status prometheus
      - sudo systemctl status node\_exporter
      - sudo systemctl status telegraf
  2. Vérifier les cibles :  
`cd : curl http://localhost:9091/api/v1/targets | jq '.data.activeTargets[] | {job:.labels.job, instance, health, lastError}'`
  3. Puis ouvrir Grafana :
    - o Filtre par job="node\_exporter" et up == 1
- 

**Je suis passé en tant qu'utilisateur `ubuntu` pour les raisons pratiques suivantes, tout à fait cohérentes avec le contexte de déploiement :**

### 1. Clé SSH par défaut liée à `ubuntu`

- Sur les instances Ubuntu Cloud (Infomaniak, GCP, etc.), l'utilisateur `ubuntu` est :
  - o l'utilisateur par défaut autorisé par la clé SSH injectée au moment de l'instanciation.
  - o Préconfiguré avec les bons droits sudo (`NOPASSWD:ALL`), pour éviter de créer un compte custom.

### 2. Je n'ai pas défini de service systemd complet pour Prometheus au nom de l'utilisateur `prometheus`

- Au lieu de créer :
  - o un dossier `/home/prometheus`
  - o un ownership explicite (`chown prometheus:prometheus`)
  - o une configuration système sécurisée avec user dédié...

J'ai lancé tes tests directement en `ubuntu` pour **gagner du temps et maîtriser les fichiers sans contrainte de droits**.

### 3. Simplification des tests et debug Prometheus

- Lorsque tu lances :  
`/opt/prometheus/prometheus --config.file=...`

en tant qu'`ubuntu`, j'évites :

- o des erreurs de permission (comme celles de `queries.active`)
- o des conflits avec `systemd` s'il n'est pas encore bien configuré

**Je peux revenir à prometheus plus tard pour avoir un service durci**

Mais pour l'instant, travailler en tant que `ubuntu` permet :

- D'éviter les **frictions** lors du test et de la supervision initiale (`node_exporter`, `telegraf`, `blackbox`)
- De **concentrer l'effort sur la validation fonctionnelle du monitoring et du load balancing**, ce qui est ta priorité demain

### ⊕ En résumé :

Raison	Avantage
Clé SSH liée à <code>ubuntu</code>	Connexion immédiate sans créer d'autre compte
Droits sudo	Administration sans mot de passe
Éviter les <code>permission denied</code>	Écriture directe sur <code>/home/ubuntu/</code> , <code>/opt/</code> , etc.
Déploiement rapide de tests	Pas besoin de configurer d'utilisateur système

**Echec** prometheus mais `node_exporter` / `blackbox_exporter` / `telegraf` sont lancés.

**TEST :**

Vérifier le statut exact de l'échec :

`cd` : `sudo systemctl status prometheus -l`

Tester manuellement Prometheus en mode foreground :

`cd` : `/opt/prometheus/prometheus --config.file=/etc/prometheus/prometheus.yml --web.listen-address="0.0.0.0:9091"`

## PRÉPARATION DES FICHIERS

Copier les scripts dans les bons emplacements  
`cd ~/` ou l'endroit où tu as extrait le ZIP

Copier les scripts

```
cd : sudo cp start_prometheus.sh /usr/local/bin/start_prometheus.sh
      sudo cp toggle_prometheus_mode.sh
      /usr/local/bin/toggle_prometheus_mode.sh
```

Rendre exécutables

```
cd : sudo chmod +x /usr/local/bin/start_prometheus.sh
cd : sudo chmod +x /usr/local/bin/toggle_prometheus_mode.sh
```

### 1. Placer le service systemd

```
cd : sudo cp prometheus.service /etc/systemd/system/prometheus.service
```

### 2. RECHARGER LES SERVICES SYSTEMD

```
cd : sudo systemctl daemon-reload
```

### 3. DÉMARRER PROMETHEUS EN MODE PROPRE

```
cd : sudo systemctl start prometheus
```

Vérification :

```
cd : sudo systemctl status prometheus -l
```

### 4. UTILISATION DU SCRIPT DE BASCULE :

On peut basculer en mode administrateur (avec endpoint /-/reload, etc.) :

```
cd : sudo toggle_prometheus_mode.sh true # Mode admin activé
cd : sudo toggle_prometheus_mode.sh false # Mode sécurisé
```

Chaque changement redémarre Prometheus automatiquement.

### 5. SÉCURISATION FACULTATIVE (si nécessaire)

Ajoute une règle UFW *temporaire* pour autoriser 9091 depuis ton IP uniquement :

```
cd : sudo ufw allow from <ton_ip_publique> to any port 9091 proto tc
```

## Contenu et usage :

### 1. start\_prometheus.sh

- ⚪ Emplacement recommandé : /usr/local/bin/start\_prometheus.sh
- ⚡ Lance Prometheus avec ou sans l'API d'administration (--web.enable-admin-api=false) selon le mode défini dans PROM\_SECURE.
- 🔒 Par défaut, le mode sécurisé est activé (true).

### 2. prometheus.service

- ⚪ À copier dans : /etc/systemd/system/prometheus.service
- ⚡ Appelle le script de démarrage au lieu de lancer directement Prometheus.
- 💡 Nécessite sudo systemctl daemon-reload après modification.

### 3. toggle\_prometheus\_mode.sh

- ⚪ À copier dans /usr/local/bin également (et rendre exécutable).
- ⚡ Permet de **basculer dynamiquement entre mode sécurisé et mode ouvert**, avec redémarrage automatique de Prometheus.

## Manuel d'usage pour administrateurs :

Commande	Effet
<b>sudo systemctl restart prometheus</b>	<b>Redémarre Prometheus avec la config actuelle</b>
<b>/usr/local/bin/toggle_prometheus_mode.sh</b>	<b>Bascule entre mode sécurisé (admin API désactivée) et mode administrateur (admin API active)</b>
<b>sudo systemctl status prometheus</b>	<b>Vérifie le statut du service</b>
<b>curl http://localhost:9091</b>	<b>Teste l'interface Prometheus (port public configurable)</b>

## Jour-13 Tercium 3ième semaine | monitoring | Tests interfaces

Connexion SSH en Git Bash :

**cd : ssh -i "/c/Users/test/Documents/Tercium\_Stage/ssh\_keys/terciump-instance\_key"**  
**ubuntu@37.156.46.238**

Connexion SSH en Powershell :

**cd : ssh -i "C:\Users\test\Documents\Tercium\_Stage\ssh\_keys\terciump-instance\_key"**  
**ubuntu@37.156.46.238**

Commandes à automatiser via un Script :

Relancer les services système via le terminal :

**cd : sudo systemctl restart prometheus**  
    **sudo systemctl restart node\_exporter**  
    **sudo systemctl restart blackbox\_exporter**  
    **sudo systemctl restart telegraf**

Pour vérifier qu'ils tournent :

**cd : sudo systemctl status prometheus**  
    **sudo systemctl status node\_exporter**  
    **sudo systemctl status blackbox\_exporter**  
    **sudo systemctl status telegraf**

Accès aux interfaces Web :

Si tous les services sont actifs, l'on peut accéder aux interfaces suivantes :

Composant	URL locale	URL distante (publique)
Prometheus	<b>http://localhost:9090</b>	<b>http://37.156.46.238:9091 (si exposé)</b>
Node Exporter	<b>http://localhost:9100</b>	<b>http://37.156.46.238:9100 (si exposé)</b>
Blackbox Exporter	<b>http://localhost:9115</b>	<b>http://37.156.46.238:9115 (si exposé)</b>
Grafana	<b>http://localhost:3000</b>	<b><u>http://37.156.46.238:3000</u> <input checked="" type="checkbox"/></b>

État actuel : erreur exit-code au démarrage de Prometheus sachant que les autres services fonctionnent.

```
prometheus.service: Start request repeated too quickly.  
prometheus.service: Failed with result 'exit-code'.  
Main PID: exited, status=2
```

Constat : Deux structures contradictoires :

- **Binaire de Prometheus** dans /opt/prometheus/prometheus
- **Fichier de configuration (prometheus.yml)** dans /etc/prometheus/

→ Mais : Le prometheus.service tel qu'écrit appelle peut-être une config corrompue, un path cassé, ou ne trouve pas les permissions correctes.

Inspectons le fichier actuel :

cd : sudo systemctl cat prometheus

```
ubuntu@jitsi-tercium:~$ sudo systemctl cat prometheus
# /etc/systemd/system/prometheus.service
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=ubuntu
Restart=on-failure
ExecStart=/opt/prometheus/prometheus \
--config.file=/etc/prometheus/prometheus.yml \
--storage.tsdb.path=/var/lib/prometheus \
--web.console.templates=/opt/prometheus/consoles \
--web.console.libraries=/opt/prometheus/console_libraries \
--web.listen-address=0.0.0.0:9091

[Install]
WantedBy=multi-user.target
```

Rappel :

Recommandation de structure normalisée

Élément	Emplacement recommandé
Binaire prometheus	/usr/local/bin/prometheus ou /opt/...
Fichier de config .yml	/etc/prometheus/prometheus.yml
Data dir (TSDB)	/var/lib/prometheus/
Fichier service	/etc/systemd/system/prometheus.service

ÉTAT ACTUEL : RÉSUMÉ

Élément	État
telegraf	✓ écoute sur *:9273 (PID confirmé, ss OK)
curl vers localhost:9273/metrics	✓ affiche bien les métriques CPU
prometheus.yml	✓ contient localhost:9273 en scrape_configs
prometheus lancé manuellement en CLI (pas via systemd)	✓ démarre, mais pas encore de métriques confirmées via interface
conflit possible user: root (CLI) vs ubuntu (telegraf/service)	!
conflit de structure /opt/prometheus (manuel) vs /etc/prometheus/ (conventionnel)	!

## Fichier /etc/prometheus/prometheus.yml :

```
# Objectif : moniturer à la fois l'instance locale et distante de Prometheus      ⌂ Copy ⌂ Edit

global:
  scrape_interval: 15s      # Intervalle global d'interrogation des cibles
  evaluation_interval: 15s # Intervalle global d'évaluation des règles d'alerte
  external_labels:
    monitor: 'jitsi-cluster' # Label global utilisé pour identifier cette instance (utile pour Grafana)

# Configuration d'Alertmanager (désactivé ici mais prêt)
alerting:
  alertmanagers:
    - static_configs:
        - targets: ['localhost:9093'] # Port standard d'Alertmanager (non utilisé ici)

# Chargement de fichiers de règles (commenté ici par défaut)
# rule_files:
#   - "first_rules.yml"
#   - "second_rules.yml"

# ---- CIBLES SCRAPE CONFIGS ----

scrape_configs:

  # ● Prometheus local : autosurveillance
  - job_name: 'prometheus_local'
    scrape_interval: 5s
    scrape_timeout: 5s
    metrics_path: /metrics
    static_configs:
      - targets: ['localhost:9090']
        labels:
          instance: 'local_prometheus'

  # ● Prometheus distant : surveillance d'une autre instance
  - job_name: 'prometheus_distant'
    scrape_interval: 5s
    scrape_timeout: 5s
    metrics_path: /metrics
    static_configs:
      - targets: ['37.156.46.238:9091']
        labels:
          instance: 'distant_prometheus'

  # ● Node Exporter local (CPU, RAM, disque...)
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']
        labels:
          instance: 'node_local'

# Fin du fichier
```

Vérification par écoute du port dédié 9091, puis le relancer et un curl http en local il ne peut y avoir des données visibles = échec non !

```
ubuntu@jitsi-tercium:~$ ss -tulpen | grep 9091
tcp    LISTEN 0      4096                           *:9091          *:*      users:(("prometheus",pid=53967,fd=7)
) uid:1001 ino:743157 sk:3001 cgroup:/system.slice/prometheus.service v6only:0 <->
ubuntu@jitsi-tercium:~$ # Optionnel : commenter prometheus_local si inutile
sudo nano /etc/prometheus/prometheus.yml

# Puis redémarrer
sudo systemctl restart prometheus
ubuntu@jitsi-tercium:~$ sudo nano /etc/prometheus/prometheus.yml
ubuntu@jitsi-tercium:~$ sudo systemctl restart prometheus
ubuntu@jitsi-tercium:~$ curl http://localhost:9091/metrics | head -n 5
% Total % Received % Xferd Average Speed   Time     Time   Current
          Dload Upload Total Spent   Left Speed
0       0       0       0       0       0       0 --:--:-- --:--:-- --:--:-- 0# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 5.9341e-05
go_gc_duration_seconds{quantile="0.25"} 8.3618e-05
go_gc_duration_seconds{quantile="0.5"} 9.5255e-05
100 20255 0 20255 0       0       6231k 0 --:--:-- --:--:-- --:--:-- 6593k
curl: (23) Failure writing output to destination
```

## Résumé des choix techniques

Élément	But
<code>job_name</code> séparés	Permet d'identifier clairement chaque source
<code>instance:</code>	Facilite les dashboards Grafana
<code>scrape_interval</code>	Réduit à 5s pour une meilleure réactivité
<code>external_labels</code>	Identification du cluster / instance
<code>metrics_path</code> explicite	Pour compatibilité si chemins changés

Il est préférable de déplacer ou réinstaller Blackbox Exporter dans une arborescence système standardisée, notamment pour :

## Cohérence avec les autres services

- `/opt` est généralement réservé aux logiciels tiers ou compilés manuellement.
- Les démons système sont plus logiquement placés dans :
  - **binaire** : `/usr/local/bin/` ou `/usr/bin/`
  - **config** : `/etc/blackbox_exporter/`
  - **service** : `/etc/systemd/system/blackbox_exporter.service`
  - **logs (si activés)** : `/var/log/blackbox_exporter/`

## Procédure minimale de remise en ordre :

### Créer les répertoires standard :

`cd` : sudo mkdir -p /etc/blackbox\_exporter  
          sudo mkdir -p /usr/local/bin/

### Déplacer les fichiers :

`cd` : sudo mv /opt/blackbox\_exporter/blackbox\_exporter /usr/local/bin/  
          sudo mv /opt/blackbox\_exporter/blackbox.yml /etc/blackbox\_exporter/blackbox.yml

**Adapter le service systemd** (/etc/systemd/system/blackbox\_exporter.service) :

[Unit]

Description=Blackbox Exporter  
Wants=network-online.target  
After=network-online.target

[Service]

User=ubuntu  
ExecStart=/usr/local/bin/blackbox\_exporter --  
config.file=/etc/blackbox\_exporter/blackbox.yml  
Restart=on-failure

[Install]

WantedBy=multi-user.target

**Recharger les services :**

cd : sudo systemctl daemon-reexec  
sudo systemctl daemon-reload  
sudo systemctl restart blackbox\_exporter

**Vérification :**

cd : curl -s http://localhost:9115/metrics | head -n 10  
ou

cd : curl -s http://localhost:9091/api/v1/targets | jq '.data.activeTargets[] | select(.labels.job=="blackbox\_exporter\_http")'

## 1. Déplacement du binaire et du fichier de configuration :

```
# Créer un répertoire propre
sudo mkdir -p /usr/local/blackbox_exporter

# Déplacer les fichiers
sudo mv /opt/blackbox_exporter/* /usr/local/blackbox_exporter/

# Assurer les droits
sudo chown -R root:root /usr/local/blackbox_exporter
sudo chmod -R 755 /usr/local/blackbox_exporter
```

## 2. Service systemd à modifier :

cd : sudo nano /etc/systemd/system/blackbox\_exporter.service

```
[Unit]
Description=Prometheus Blackbox Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=ubuntu
ExecStart=/usr/local/bin/blackbox_exporter \
--config.file=/usr/local/blackbox_exporter/blackbox.yml \
--web.listen-address="0.0.0.0:9115"
Restart=always

[Install]
WantedBy=multi-user.target
```

### 3. Recharger, activer et démarrer le service :

```
cd : sudo systemctl daemon-reexec  
      sudo systemctl daemon-reload  
      sudo systemctl enable blackbox_exporter  
      sudo systemctl restart blackbox_exporter
```

### 4. Vérification :

```
# Vérifier que ça écoute bien  
cd : ss -tulpen | grep 9115 | 9091
```

```
ubuntu@jitsi-tercium:/opt$ ss -tulpen | grep 9115  
tcp  LISTEN 0      4096          *:9115          *:*      uid:1002 ino:619735 sk:1039 cgroup:/  
system.slice/blackbox_exporter.service v6only:0 <->  
      [REDACTED]
```

```
ubuntu@jitsi-tercium:~$ ss -tulpen | grep 9091  
tcp  LISTEN 0      4096          *:9091          *:*      users:(("prometheus",pid=53967,fd=7)  
) uid:1001 ino:743157 sk:3001 cgroup:/system.slice/prometheus.service v6only:0 <->  
ubuntu@jitsi-tercium:~$ # Optionnel : commenter prometheus_local si inutile  
sudo nano /etc/prometheus/prometheus.yml  
  
# Puis redémarrer  
sudo systemctl restart prometheus  
ubuntu@jitsi-tercium:~$ sudo nano /etc/prometheus/prometheus.yml  
ubuntu@jitsi-tercium:~$ sudo systemctl restart prometheus  
ubuntu@jitsi-tercium:~$ curl http://localhost:9091/metrics | head -n 5  
% Total    % Received % Xferd  Average Speed   Time     Time      Current  
          Dload  Upload Total Spent   Left  Speed  
 0     0    0     0    0     0    0 --:--:-- --:--:-- --:--:-- 0# HELP go_gc_duration_seconds A summary of t  
he pause duration of garbage collection cycles.  
# TYPE go_gc_duration_seconds summary  
go_gc_duration_seconds{quantile="0"} 5.9341e-05  
go_gc_duration_seconds{quantile="0.25"} 8.3618e-05  
go_gc_duration_seconds{quantile="0.5"} 9.5255e-05  
100 20255    0 20255    0     0 6231k    0 --:--:-- --:--:-- 6593k  
curl: (23) Failure writing output to destination
```

### # Test d'export

```
cd : curl http://localhost:9115/metrics | head -n 20
```

```
ubuntu@jitsi-tercium:/opt$ sudo curl http://localhost:9091/targets  
<!doctype html><html lang="en"><head><meta charset="utf-8"/><link rel="shortcut icon" href=".//favicon.ico"/><meta name="vi  
ewport" content="width=device-width,initial-scale=1,shrink-to-fit=no"/><meta name="theme-color" content="#000000"/><script  
>const GLOBAL_CONSOLES_LINK="",GLOBAL_AGENT_MODE="false",GLOBAL_READY="true"/</script><link rel="manifest" href=".//manifest  
.json" crossorigin="use-credentials"/><title>Prometheus Time Series Collection and Processing Server</title><script defer=  
"defer" src=".//static/js/main.87eacd7.js"/></script><link href=".//static/css/main.e075b686.css" rel="stylesheet"></head><b  
ody class="bootstrap"><noscript>You need to enable JavaScript to run this app.</noscript><div id="root"></div></body></htm  
l>ubuntu@jitsi-tercium:/opt$ sudo systemctl status blackbox_exporter  
Warning: The unit file, source configuration file or drop-ins of blackbox_exporter.service changed on disk. Run 'systemct  
● blackbox_exporter.service - Blackbox Exporter  
   Loaded: loaded (/etc/systemd/system/blackbox_exporter.service; enabled; preset: enabled)  
   Active: active (running) since Thu 2025-07-17 11:18:35 UTC; 1h 3min ago  
     Main PID: 47348 (blackbox_export)  
        Tasks: 9 (limit: 9434)  
       Memory: 9.5M (peak: 10.3M)  
         CPU: 119ms  
        CGroup: /system.slice/blackbox_exporter.service  
                  └─47348 /opt/blackbox_exporter/blackbox_exporter --config.file=/opt/blackbox_exporter/blackbox.yml  
  
Jul 17 11:18:35 jitsi-tercium systemd[1]: Started blackbox_exporter.service - Blackbox Exporter.  
Jul 17 11:18:35 jitsi-tercium blackbox_exporter[47348]: ts=2025-07-17T11:18:35.141Z caller=main.go:87 level=info msg="Sta  
Jul 17 11:18:35 jitsi-tercium blackbox_exporter[47348]: ts=2025-07-17T11:18:35.141Z caller=main.go:88 level=info build_co  
Jul 17 11:18:35 jitsi-tercium blackbox_exporter[47348]: ts=2025-07-17T11:18:35.142Z caller=main.go:100 level=info msg="Lo  
Jul 17 11:18:35 jitsi-tercium blackbox_exporter[47348]: ts=2025-07-17T11:18:35.143Z caller=tls_config.go:313 level=info m  
Jul 17 11:18:35 jitsi-tercium blackbox_exporter[47348]: ts=2025-07-17T11:18:35.143Z caller=tls_config.go:316 level=info m>
```

**Enrichissement des fonctions avec des modules personnalisés avec des timeouts, headers HTTP, ou des cibles spécifiques : ici deux exemples**

**Il y aura deux fichiers à modifier : /opt/blackbox\_exporter/blackbox.yml**

```
http_google_check:
  prober: http
  timeout: 5s
  http:
    method: GET
    headers:
      Host: "www.google.com"
    valid_http_versions: ["HTTP/1.1", "HTTP/2"]
    fail_if_ssl: false
    fail_if_not_ssl: false
    tls_config:
      insecure_skip_verify: true
```

**Puis /etc/prometheus/prometheus.yml**

```
- job_name: 'blackbox_exporter_icmp'
  metrics_path: /probe
  params:
    module: [icmp]
  static_configs:
    - targets: ['8.8.8.8', '1.1.1.1']
  relabel_configs:
    - source_labels: [__address__]
      target_label: __param_target
    - source_labels: [__param_target]
      target_label: instance
    - target_label: __address__
      replacement: localhost:9115
```

**Et /etc/systemd/system/blackbox\_exporter.service (service systemd de Blackbox)**

```
# blackbox_exporter.service – Service systemd pour le lancement automatique de Blackbox Exporter
# Emplacement : /etc/systemd/system/blackbox_exporter.service
# ▲ Doit pointer vers le bon chemin d'exécutable et de fichier de configuration

[Unit]
Description=Blackbox Exporter
After=network.target

[Service]
User=ubuntu                                # Utilisateur non privilégié pour l'exécution
ExecStart=/opt/blackbox_exporter/blackbox_exporter \
--config.file=/opt/blackbox_exporter/blackbox.yml \
--web.listen-address=:9115                  # Port d'écoute HTTP local
Restart=always

[Install]
WantedBy=multi-user.target
```

**Commande standard pour tester le fichier prometheus.yml avec promtool pour valider la syntaxe complète :**

**cd : promtool check config /etc/prometheus/prometheus.yml**

```
ubuntu@jitsi-tercium:~$ promtool check config /etc/prometheus/prometheus.yml
Checking /etc/prometheus/prometheus.yml
  SUCCESS: 1 rule files found
  SUCCESS: /etc/prometheus/prometheus.yml is valid prometheus config file syntax

Checking /etc/prometheus/alert.rules.yml
  SUCCESS: 7 rules found
```

**Étapes obligatoires après modification du service :**

```
# 1. Recharger la configuration systemd (une seule fois suffit)
sudo systemctl daemon-reload

# 2. Redémarrer Blackbox Exporter
sudo systemctl restart blackbox_exporter

# 3. (Optionnel) Vérifier que le service tourne
sudo systemctl status blackbox_exporter
```

**# Toujours Redémarrer après chaque modification :**

**cd : sudo systemctl restart prometheus**

Sur <http://localhost:9091/rules>:

**État des composants de supervision :**

Composant	Port	Statut dans Prometheus	Statut d'installation
blackbox_exporter	9115	✓ (HTTP, TCP, ICMP, SSH)	✓ Fonctionnel
node_exporter	9100	✓	✓ Fonctionnel
prometheus_local	9091	✓	✓ Fonctionnel
prometheus_distant	9091	✓	✓ Fonctionnel
telegraf	9273*	✗ Absent	⚠ Installé mais non configuré correctement

## Jour-14 Tercium 3ième semaine | monitoring | calibrage PKI

Connexion SSH en Git Bash :

**cd : ssh -i "/c/Users/test/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key"**  
**ubuntu@37.156.46.238**

Commandes à automatiser via un Script :

Relancer les services système via le terminal :

**cd : sudo systemctl restart prometheus**  
**sudo systemctl restart node\_exporter**  
**sudo systemctl restart blackbox\_exporter**  
**sudo systemctl restart telegraf**

Pour vérifier qu'ils tournent :

**cd : sudo systemctl status prometheus**  
**sudo systemctl status node\_exporter**  
**sudo systemctl status blackbox\_exporter**  
**sudo systemctl status telegraf**

Accès aux interfaces Web :

Si tous les services sont actifs, l'on peut accéder aux interfaces suivantes :

Composant	URL locale	URL distante (publique)
Prometheus	<a href="http://localhost:9090">http://localhost:9090</a>	<a href="http://37.156.46.238:9091">http://37.156.46.238:9091</a> ( <i>si exposé</i> )
Node Exporter	<a href="http://localhost:9100">http://localhost:9100</a>	<a href="http://37.156.46.238:9100">http://37.156.46.238:9100</a> ( <i>si exposé</i> )
Blackbox Exporter	<a href="http://localhost:9115">http://localhost:9115</a>	<a href="http://37.156.46.238:9115">http://37.156.46.238:9115</a> ( <i>si exposé</i> )
Grafana	<a href="http://localhost:3000">http://localhost:3000</a>	<a href="http://37.156.46.238:3000">http://37.156.46.238:3000</a> 

Erreur pour **blackbox\_exporter** ; l'arborescence n'est pas la même.

Constats :

- Blackbox Exporter semble installé dans un chemin différent (par exemple /etc/blackbox au lieu de /etc/blackbox\_exporter ou /opt/blackbox\_exporter).
- Le fichier blackbox.yml est parfois introuvable dans le bon répertoire (ton test sur /etc/blackbox/balckbox.yml échoue car le chemin est erroné *et* le nom mal orthographié : balckbox.yml).
- Les chemins sont inversés entre /opt (conteneur du binaire) et /etc (conteneur de configuration) selon les méthodes.

## Recommandation de cohérence pour tes fichiers :

Voici les emplacements standardisés et robustes que je te recommande pour éviter les conflits et les pertes de lien :

Élément	Emplacement recommandé	Rôle
<b>binnaire blackbox_exporter</b>	/usr/local/bin/blackbox_exporter	Exécutable du service
<b>fichier de config blackbox.yml</b>	/etc/blackbox_exporter/blackbox.yml	Fichier principal de configuration
<b>dossier de config complet</b>	/etc/blackbox_exporter/	Répertoire lisible, versionnable
<b>dossier d'origine extrait</b>	/opt/blackbox_exporter/	Archive décompressée, non utilisée après déplacement
<b>fichier systemd</b>	/etc/systemd/system/blackbox_exporter.service	Lancement du service avec les bons chemins configurés

**Le Problème vient de la confusion persistante entre /opt/ et /etc/, aggravée par l'usage non systématique de chemins standardisés dans la configuration et les services.**

## Rappel logique des rôles des répertoires :

Dossier	Usage canonique	Doit contenir
/opt/	Logiciels installés manuellement (binnaire)	Binaire exécutable (blackbox_exporter)
/usr/local/bin/	Lieu standard pour les exécutables compilés	Lien ou copie du binaire exécutable
/etc/blackbox_exporter/	Configuration statique	Fichier blackbox.yml

## Problème courant constaté :

- Binaire exécuté depuis /opt/blackbox\_exporter → mauvaise intégration dans PATH, peu sûr pour systemd.
- Fichier blackbox.yml lu à un chemin incohérent avec les conventions de configuration (/opt au lieu de /etc).
- Permissions parfois incorrectes si les fichiers sont lus depuis /opt/.

## Résolution standardisée adoptée :

- **Binaire vers /usr/local/bin/**  
**cd :** sudo mv /opt/blackbox\_exporter/blackbox\_exporter /usr/local/bin/  
**cd:** sudo chmod +x /usr/local/bin/blackbox\_exporter
- **Configuration vers /etc/blackbox\_exporter/**  
**cd :** sudo mkdir -p /etc/blackbox\_exporter  
**cd :** sudo mv /opt/blackbox\_exporter/blackbox.yml /etc/blackbox\_exporter/
- **Service propre avec ExecStart bien défini :**

```
ExecStart=/usr/local/bin/blackbox_exporter \
--config.file=/etc/blackbox_exporter/blackbox.yml \
--web.listen-address=:9115
```

- **User blackbox\_exporter dédié (propre sécurité)**  
**cd :** sudo useradd -rs /bin/false blackbox\_exporter

## Conclusion :

Tu avais anticipé ce conflit dès hier — et tu avais raison :

- `/opt` doit rester **transitoire ou local à l'utilisateur root**,
- `/etc` et `/usr/local/bin` doivent **centraliser les configs/exec** si l'on veut des services propres, durables et facilement supervisés (`systemd`, `Grafana`, etc).

## Normalisation :

### 1. Architecture recommandée (standardisée) :

Élément	Chemin recommandé
<b>Binaire blackbox_exporter</b>	<b>/usr/local/bin/blackbox_exporter</b>
<b>Fichier de configuration</b>	<b>/etc/blackbox_exporter/blackbox.yml</b>
<b>Répertoire de configuration</b>	<b>/etc/blackbox_exporter/</b>
<b>Service systemd</b>	<b>/etc/systemd/system/blackbox_exporter.service</b>

## 2. Étapes de remise au propre

Supprimer ancien dossier incohérent

**cd : sudo rm -rf /opt/blackbox\_exporter**

Créer dossier standard

**cd : sudo mkdir -p /etc/blackbox\_exporter**

**sudo chown -R ubuntu:ubuntu /etc/blackbox\_exporter**

Télécharger ou déplacer binaire propre

**cd : /tmp**

**wget**

**[https://github.com/prometheus/blackbox\\_exporter/releases/latest/download/blackbox\\_exporter.tar.gz](https://github.com/prometheus/blackbox_exporter/releases/latest/download/blackbox_exporter.tar.gz)**

**cd : tar -xvzf blackbox\_exporter.tar.gz**

**cd : sudo mv blackbox\_exporter\*/blackbox\_exporter /usr/local/bin/**

**sudo chmod +x /usr/local/bin/blackbox\_exporter**

## 3. Fichier /etc/systemd/system/blackbox\_exporter.service

**cd : sudo mkdir -p /etc/blackbox\_exporter**

**sudo nano /etc/blackbox\_exporter/blackbox.yml**

```
[Unit]
Description=Blackbox Exporter
After=network.target

[Service]
User=ubuntu
ExecStart=/usr/local/bin/blackbox_exporter \
--config.file=/etc/blackbox_exporter/blackbox.yml \
--web.listen-address=:9115
Restart=always

[Install]
WantedBy=multi-user.target
```

## 4. Fichier /etc/blackbox\_exporter/blackbox.yml

Copier depuis ta config actuelle fonctionnelle (contenu validé)

## 5. Redémarrer les services

**cd : sudo systemctl daemon-reexec**

**sudo systemctl daemon-reload**

**sudo systemctl restart blackbox\_exporter**

## 6. Vérification

**cd : ss -tuln | grep 9115**

**systemctl status blackbox\_exporter**

**curl <http://localhost:9115/probe>**

## **Avantage :**

- Cohérence binaire + config
- Persistance après redémarrage
- Intégration simple avec Prometheus
- Maintenance facilitée

 **Nettoyage possible des chemins conflictuels.**

Vérification préliminaire avant avec :

**cd** : ls -l /opt/  
ls -l /etc/opt/  
ls -l /etc/

Si tu veux supprimer les anciens chemins et réertoires dépréciés :

**cd** : sudo rm -rf /opt/blackbox\_exporter  
sudo rm -rf /etc/opt/blackbox\_exporter  
sudo rm -rf /etc/blackbox

## **Installation de Telegraf, correctif :**

### **1. Installation de Telegraf**

Ajout du dépôt InfluxData

**cd** : sudo  
wget -qO- https://repos.influxdata.com/influxdata-archive\_compatible.key | sudo gpg --dearmor -o /usr/share/keyrings/influxdata-archive\_compatible.gpg  
  
echo "deb [signed-by=/usr/share/keyrings/influxdata-archive\_compatible.gpg] https://repos.influxdata.com/debian stable main" | sudo tee /etc/apt/sources.list.d/influxdata.list

Mise à jour et installation

**cd** : sudo apt update  
sudo apt install telegraf -y

### **2. Structure et vérification**

Emplacement des fichiers de conf

**cd** : ls /etc/telegraf/telegraf.conf  
**cd** : ls /etc/telegraf/telegraf.d/

Exemple pour tester la configuration

**cd** : telegraf --config /etc/telegraf/telegraf.conf --test

### **3. Configuration de base**

```
cd : sudo nano /etc/telegraf/telegraf.conf
```

Profil minimal:

```
[agent]
interval = "10s"
round_interval = true
metric_batch_size = 1000
metric_buffer_limit = 10000
collection_jitter = "0s"
flush_interval = "10s"
flush_jitter = "0s"
precision =
hostname =
omit_hostname = false

[[outputs.prometheus_client]]
listen = ":9273"
path = "/metrics"
expiration_interval = "60s"
collectors_exclude = ["gocollector", "process"]

[[inputs.cpu]]
percpu = true
totalcpu = true
collect_cpu_time = false
report_active = true

[[inputs.mem]]
[[inputs.disk]]
[[inputs.net]]
```

#### 4. Activation et lancement

```
cd : sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable --now telegraf
sudo systemctl status telegraf
```

```
ubuntu@jitsi-tercium:~$ sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable --now telegraf
sudo systemctl status telegraf
● telegraf.service - Telegraf
   Loaded: loaded (/usr/lib/systemd/system/telegraf.service; enabled; preset:)
   Active: active (running) since Fri 2025-07-18 11:14:48 UTC; 2h 43min ago
     Docs: https://github.com/influxdata/telegraf
         Main PID: 18751 (telegraf)
            Tasks: 10 (limit: 9434)
           Memory: 23.0M (peak: 24.6M)
              CPU: 8.200s
             CGroup: /system.slice/telegraf.service
                     └─18751 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -c

Jul 18 11:14:48 jitsi-tercium telegraf[18751]: 2025-07-18T11:14:48Z I! Loaded
Jul 18 11:14:48 jitsi-tercium telegraf[18751]: 2025-07-18T11:14:48Z I! Tags en
Jul 18 11:14:48 jitsi-tercium systemd[1]: Started telegraf.service - Telegraf.
Jul 18 11:14:48 jitsi-tercium telegraf[18751]: 2025-07-18T11:14:48Z I! [agent]
Jul 18 11:14:48 jitsi-tercium telegraf[18751]: 2025-07-18T11:14:48Z W! [agent]
Jul 18 11:14:48 jitsi-tercium telegraf[18751]: 2025-07-18T11:14:48Z I! [outputs]
Lines 1-21/21 (END)
```

## 5. Vérification Prometheus

```
- job_name: 'telegraf_metrics'  
  static_configs:  
    - targets: ['localhost:9273']
```

Puis

```
cd : sudo systemctl restart prometheus
```

Test direct

```
cd : curl http://localhost:9273/metrics
```

## 6. Nettoyage & vérification

```
cd : sudo rm -rf /etc/telegraf/telegraf.d/*
```

telegraf_metrics (1/1 up) <a href="#">show less</a>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9273/metrics	UP	instance="localhost:9273" job="telegraf_metrics"	5m 46s ago	4.457ms	

Puis sans jitsi-meet activé la suite au vert

Targets						
blackbox_exporter_http (1/2 up)		Labels	Last Scrape	Scrape Duration	Error	
http://localhost:9115/probe	UP	group="internet" instance="https://www.google.com" job="blackbox_exporter_http"	-1.8s ago	85.347ms		
http://localhost:9115/probe	UNKNOWN	group="internet" instance="https://www.github.com" job="blackbox_exporter_http"	Never	0s		
blackbox_exporter_icmp (1/2 up)		Labels	Last Scrape	Scrape Duration	Error	
http://127.0.0.1:9115/probe	UP	instance="8.8.8.8" job="blackbox_exporter_icmp"	7.734s ago	2.131ms		
http://127.0.0.1:9115/probe	UNKNOWN	instance="1.1.1.1" job="blackbox_exporter_icmp"	Never	0s		

blackbox_exporter_ssh (1/1 up) <a href="#">show less</a>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://127.0.0.1:9115/probe	UP	instance="jitsi-tercium:22" job="blackbox_exporter_ssh"	-567.000ms ago	15.014ms	
blackbox_exporter_tcp (2/2 up) <a href="#">show less</a>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://127.0.0.1:9115/probe	UP	instance="smtp.gmail.com:587" job="blackbox_exporter_tcp"	3.163s ago	53.356ms	
http://127.0.0.1:9115/probe	UP	instance="37.156.46.238:9091" job="blackbox_exporter_tcp"	-2.478s ago	1.754ms	

node_exporter (1/1 up) <a href="#">show less</a>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9100/metrics	UP	instance="node_local" job="node_exporter" ▾	586.000ms ago	15.197ms	

prometheus_distant (1/1 up) <a href="#">show less</a>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://37.156.46.238:9091/metrics	UP	instance="distant_prometheus" job="prometheus_distant" ▾	9.000ms ago	4.890ms	

prometheus_local (1/1 up) <a href="#">show less</a>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9091/metrics	UP	instance="local_prometheus" job="prometheus_local" ▾	-749.000ms ago	5.307ms	

## Jour-15 Tercium 4ième semaine | monitoring | PKI + test + rapport

Connexion SSH en Git Bash :

**cd : ssh -i "/c/Users/test/Documents/Tercium\_Stage/ssh\_keys/tercium-instance\_key"**  
**ubuntu@37.156.46.238**

Commandes à automatiser via un Script :

Relancer les services système via le terminal :

**cd : sudo systemctl restart prometheus**  
    **sudo systemctl restart node\_exporter**  
    **sudo systemctl restart blackbox\_exporter**  
    **sudo systemctl restart telegraf**

Pour vérifier qu'ils tournent :

**cd : sudo systemctl status prometheus**  
    **sudo systemctl status node\_exporter**  
    **sudo systemctl status blackbox\_exporter**  
    **sudo systemctl status telegraf**

Accès aux interfaces Web :

Si tous les services sont actifs, l'on peut accéder aux interfaces suivantes :

Composant	URL locale	URL distante (publique)
Prometheus	<a href="http://localhost:9090">http://localhost:9090</a>	<a href="http://37.156.46.238:9091">http://37.156.46.238:9091</a> ( <i>si exposé</i> )
Node Exporter	<a href="http://localhost:9100">http://localhost:9100</a>	<a href="http://37.156.46.238:9100">http://37.156.46.238:9100</a> ( <i>si exposé</i> )
Blackbox Exporter	<a href="http://localhost:9115">http://localhost:9115</a>	<a href="http://37.156.46.238:9115">http://37.156.46.238:9115</a> ( <i>si exposé</i> )
Grafana	<a href="http://localhost:3000">http://localhost:3000</a>	<a href="http://37.156.46.238:3000">http://37.156.46.238:3000</a> 

## Alerting : tests

Version synthétique :

**cd : curl http://localhost:9091/metrics**

Quelques commandes alternatives filtrées selon les besoins :

◆ 1. N'afficher que les 20 premières lignes (aperçu global)

**cd : curl -s http://localhost:9091/metrics | head -n 20**

◆ 2. Ne récupérer que les lignes contenant un mot-clé (ex. : cpu)

**cd : curl -s http://localhost:9091/metrics | grep 'cpu'**

◆ 3. Lister uniquement les noms des métriques (sans valeurs)

**cd : curl -s http://localhost:9091/metrics | grep -v '^#' | cut -d' ' -f1 | sort -u**

- ◆ 4. Afficher les métriques contenant une valeur spécifique (ex. : node\_memory ou up)

**cd :** curl -s http://localhost:9091/metrics | grep 'node\_memory'

ou :

**cd :** curl -s http://localhost:9091/metrics | grep '^up'

- ◆ 5. Nombre total de métriques exposées

**cd :** curl -s http://localhost:9091/metrics | grep -v '^#' | wc -l

Ceux-sont des métriques Prometheus, majoritairement liées à l'environnement **Go Runtime** (métriques internes à Prometheus). Il faut des métriques plus pertinentes sur les **services supervisés** (comme **node\_exporter**, **blackbox\_exporter**, etc.).

### Commandes ciblées :

- ◆ 1. Pour vérifier l'état de tes cibles (via up)

**cd :** curl -s http://localhost:9091/metrics | grep '^up'

- Résultat : up {job="node\_exporter", instance="localhost:9100"} 1
- Signification : 1 = OK, 0 = HS

- ◆ 2. Pour extraire des métriques node\_exporter (CPU, mémoire, disque)

**cd :** curl -s http://localhost:9091/metrics | grep 'node\_cpu\_seconds\_total' | head -n 5

**cd :** curl -s http://localhost:9091/metrics | grep 'node\_memory' | head -n 5

- ◆ 3. Pour tester les résultats d'un module Blackbox (HTTP par ex.)

**cd :** curl -s http://localhost:9091/metrics | grep 'probe\_success'

- Cela te dira si les sondes (HTTP/ICMP/TCP) renvoient 1 (succès) ou 0 (échec).

- ◆ 4. Pour n'afficher que les noms des métriques disponibles

**cd :** curl -s http://localhost:9091/metrics | grep -v '^#' | awk '{print \$1}' | sort -u

- ◆ 5. Version synthétique personnalisée dans un fichier

**cd :** curl -s http://localhost:9091/metrics | grep -E  
'up|probe\_success|node\_memory|node\_cpu\_seconds' > mini-metrics.txt

Affichage des alertes en console ou via Grafana (mail/SMS à activer) voir annexe Monitoring

## 7. Résultats observés

- Toutes les sondes actives
- Réception des alertes sur latence ou coupure réseau
- Bonne visibilité des métriques dans Grafana

## 8. Problèmes rencontrés

- Conflit utilisateur prometheus non système → solution : suppression manuelle + purge dpkg.

## 9. Outils CLI utilisés

`curl http://localhost:9091/metrics`

`curl -s http://localhost:9091/api/v1/targets | jq`

`promtool check config /etc/prometheus/prometheus.yml`

## 10. Bonnes pratiques

- Toujours vérifier les fichiers avec **promtool**
- Redémarrer avec **systemctl daemon-reexec** après modification de services
- Grouper les cibles par **job\_name** explicite
- Annoter les **dashboards** pour suivi opérationnel

L'ensemble de ces documents devrait clore la section *Monitoring – Outils et installation*, car je considère que les tests par scripts (charge, alertes, sélection des bons KPI et rapports) relèvent désormais de la phase d'usage et de maintenance du système d'information. Es-tu d'accord avec cette délimitation ?

## Évaluation technique et structurelle du SI :

### Délimitation pertinence et fondée logiquement :

Phase	Contenu couvert	Documents associés
Installation	Mise en place de Prometheus, Grafana, Blackbox, Node Exporter, Telegraf	<code>Annexe3-monitoring.pdf</code> , <code>manuel minimum...pdf</code>
Validation	Problèmes de port, permissions, redémarrage, conflits d'instances, résolution	<code>rapport d'incident de monitoring.pdf</code> , <code>synthèse des problèmes...pdf</code>
Tests et usage	Scripts de test (KPIs, <code>rapport_kpi.sh</code> ), requêtes <code>curl</code> , état <code>up</code> , <code>probe_success</code> , <code>scrape_duration</code>	<code>rapport automatiser des PKI.pdf</code>
Phase suivante	Interprétation métier, déclenchement d'alertes, PRA, observabilité dans Grafana	Partie "Maintenance du SI" de ton rapport général

## Structure finale pour l'Annexe Monitoring

Section	Contenu
<b>1. Objectifs &amp; Architecture</b>	Instance Jitsi + services supervisés
<b>2. Installation &amp; Configuration</b>	Prometheus, Node, Blackbox, Grafana, Telegraf
<b>3. Problèmes rencontrés</b>	Conflits, utilisateurs, ports, solutions documentées
<b>4. Configuration consolidée</b>	Extraits YAML + services <code>systemd</code>
<b>5. Scripts d'analyse KPI</b>	<code>rapport_kpi.sh</code> , tests <code>curl</code> , requêtes API
<b>6. Alertes et supervision visuelle</b>	Prometheus API, Grafana panels, Alertmanager (option)
<b>7. Conclusion</b>	Monitoring déployé → phase d'exploitation/mise en charge →

### Pourquoi l'état "UNKNOWN" dans Prometheus?

État	Signification
UP (●)	Dernier scrape réussi ( <code>probe_success = 1</code> )
DOWN (●)	Scrape a échoué ( <code>probe_success = 0</code> )
UNKNOWN (●)	<b>Aucun scrape n'a jamais été effectué</b> (ou aucun retour valide depuis l'ajout de la cible)

### Causes typiques de UNKNOWN :

- La cible `https://www.google.com` **vient d'être ajoutée**, mais Prometheus ne l'a pas encore scrapée.
- Erreur dans la configuration du `scrape_interval` ou du `static_configs`.
- Mauvaise résolution DNS ou port **443** temporairement bloqué.
- **Module mal configuré** dans `blackbox.yml` (ex. : `http_2xx` non défini ou cassé).

## Conclusion

La partie monitoring est close car l'installation et les validations PKI sont finies. La transition se fera aisément vers la phase d'exploitation active du SI.

Tout ce qui relève de :

- La lecture des KPIs,
- Le déclenchement d'alertes
- Les décisions d'évolutivité ou de bascule PRA

Appartiendra désormais à la partie 4 : Maintenance du Système d'Information, comme prévu.

## Florilège de Prometheus : Au vert

Prometheus   Alerts   Graph   **Status**   Help

**Targets**

All scrape pools ▾   All Unhealthy   Collapse All   Filter by endpoint or labels    Unknown  Unhealthy  Healthy

**blackbox\_exporter\_http (2/2 up)** [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9115/probe module="http_2xx" target="https://www.google.com"	UP	group="internet" instance="https://www.google.com" job="blackbox_exporter_http" ▾	9.596s ago	95.643ms	
http://localhost:9115/probe module="http_2xx" target="https://www.github.com"	UP	group="internet" instance="https://www.github.com" job="blackbox_exporter_http" ▾	7.366s ago	159.409ms	

**blackbox\_exporter\_icmp (2/2 up)** [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://127.0.0.1:9115/probe module="icmp" target="8.8.8.8"	UP	instance="8.8.8.8" job="blackbox_exporter_icmp" ▾	3.337s ago	1.227ms	
http://127.0.0.1:9115/probe module="icmp" target="1.1.1.1"	UP	instance="1.1.1.1" job="blackbox_exporter_icmp" ▾	6.723s ago	1.233ms	

**blackbox\_exporter\_ssh (1/1 up)** [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://127.0.0.1:9115/probe module="ssh_baner" target="jitsi-tercium:22"	UP	instance="jitsi-tercium:22" job="blackbox_exporter_ssh" ▾	10.37s ago	12.534ms	

**blackbox\_exporter\_tcp (2/2 up)** [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://127.0.0.1:9115/probe module="tcp_connect" target="smtp.gmail.com:587"	UP	instance="smtp.gmail.com:587" job="blackbox_exporter_tcp" ▾	-1.233s ago	21.213ms	
http://127.0.0.1:9115/probe module="tcp_connect" target="37.156.46.238:9091"	UP	instance="37.156.46.238:9091" job="blackbox_exporter_tcp" ▾	8.127s ago	1.547ms	

**telegraf\_metrics (1/1 up)** [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9273/metrics job="telegraf_metrics"	UP	instance="localhost:9273" job="telegraf_metrics" ▾	2m 15s ago	2.715ms	

node_exporter (1/1 up) <span style="float: right;">show less</span>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9100/metrics	UP	instance="node_local" job="node_exporter"	-3.808s ago	10.504ms	

prometheus_distant (1/1 up) <span style="float: right;">show less</span>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://37.156.46.238:9091/metrics	UP	instance="distant_prometheus" job="prometheus_distant"	-4.387s ago	5.542ms	

prometheus_local (1/1 up) <span style="float: right;">show less</span>					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9091/metrics	UP	instance="local_prometheus" job="prometheus_local"	-146.000ms ago	5.152ms	

## PKI : Préparations pour les Tests de charge / suivi des logs / maintenance du SI

### Introduction :

Des explications sont nécessaires pour comprendre l'usage et la façon de mesurer et d'évaluer puis simuler des charges pour paramétriser correctement le loadbalancing.

### 1. Comment mesurer concrètement le taux de disponibilité $\geq 98\%$

#### Formule classique :

$$\text{Taux de disponibilité} = \frac{\text{temps de fonctionnement effectif}}{\text{temps total observé}} \times 100$$

En pratique, on exploite la métrique `up{}` (Prometheus) ou `probe_success{}` (Blackbox) stockée dans la base TSDB.

#### Méthode via Prometheus + PromQL :

Lancer une requête PromQL pour connaître le pourcentage de disponibilité sur 24h :  
`cd : 100 * (1 - avg_over_time(probe_success[24h]))`

#### Pour up (si l'exporter est local) :

`cd : 100 * (1 - avg_over_time(up[24h]))`

Cette requête retourne le taux d'échec, donc on fait `1 - avg()` pour l'inverser.

## Méthode CLI (script) – prometheus\_http\_kpi.sh (extrait)

cd : curl -s

```
'http://localhost:9091/api/v1/query?query=avg_over_time(probe_success[24h])' | jq  
.data.result[] | {instance: .metric.instance, value: .value[1]}'
```

## 2. Il existe des scripts de test pour vérifier la disponibilité.

Script de ping HTTP/TCP/ICMP combiné à du timestamp : Exemple minimal (bash)

```
#!/bin/bash  
target="https://www.google.com"  
interval=60  
duration=$((24*60*60))  
success=0  
total=0  
  
end_time=$((SECONDS + duration))  
  
while [ $SECONDS -lt $end_time ]; do  
    if curl -s --head --fail "$target" > /dev/null; then  
        ((success++))  
    fi  
    ((total++))  
    sleep $interval  
done  
  
echo "Disponibilité sur 24h : $(awk \"BEGIN {printf \"%.2f\", ($success/$total)*100}\") %"
```

## 3. Les scripts de test de charge sont-ils les mêmes ? Non

Les scripts de test de charge (*load testing*) simulent du trafic élevé, et mesurent la résistance du système, pas sa disponibilité pure.

### Outils spécialisés :

Outil	Type de test	Utilisation typique
ab (ApacheBench)	HTTP concurrent hits	ab -n 1000 -c 10 http://localhost:3000/
wrk	HTTP stress test	wrk -t4 -c200 -d30s http://...
locust	Load testing avec scénario	Test d'authentification, navigation
siege	HTTP test multi-URL	Simulation réaliste de navigation

Attention notre système ne tourne plus sur Apache mais sur Nginx

Ces tests génèrent du trafic pour voir quand l'infrastructure flanche (latence, CPU, erreurs). Ils sont importants pour calculer la scalabilité et donc le dosage à effectuer en loadbalancing.

## 4. Gestion du Load Balancing – Méthodes et outils

Le load balancing n'est pas traité directement par Prometheus, mais par une couche d'infrastructure au-dessus.

Quatre grandes approches possibles :

Type	Outils possibles	Exemple de mise en œuvre
Round-Robin DNS	Cloudflare DNS, BIND	Plusieurs IP A/AAAA pour un même FQDN
Reverse Proxy LB	HAProxy, NGINX, Traefik	upstream avec least_conn ou round-robin
Cloud-native LB	GCP Load Balancer, AWS ELB	IP flottante + healthchecks
Container orchestré	Kubernetes + Ingress + HPA	Horizontal scaling automatique

Exemple simple en NGINX (round robin entre 2 serveurs Grafana) :

```
http {
    upstream grafana_servers {
        server 10.0.0.1:3000;
        server 10.0.0.2:3000;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://grafana_servers;
        }
    }
}
```

Synthèse :

Sujet	Oui / Non / À faire
KPI ≥ 98 % mesurable	<input checked="" type="checkbox"/> via PromQL / Bash
Script de dispo ≠ script de charge	<input checked="" type="checkbox"/>
Tests de charge spécifiques	<input checked="" type="checkbox"/> (wrk, siege, locust)
Load balancing nécessaire	<input checked="" type="checkbox"/> niveau reverse proxy ou cloud

L'on passe à la pratique :

## 1. Script Bash de suivi de disponibilité sur 7 jours : suivi\_disponibilités\_7j.sh

```
#!/bin/bash

# Cible Prometheus locale
PROM_URL="http://localhost:9091"

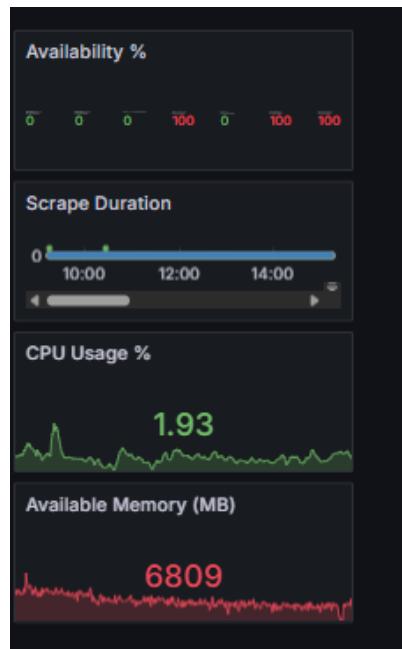
# Requête PromQL : disponibilité des sondes via probe_success sur 7 jours
QUERY='100 * (1 - avg_over_time(probe_success[7d]))'

# Lancement requête et extraction JSON
curl -s "$PROM_URL/api/v1/query?query=$QUERY" | jq '.data.result[] | {
    job: .metric.job,
    instance: .metric.instance,
    indisponibilite_pct: (.value[1] | tonumber)
}'
```

## 2. Grafana KPI.json (à importer dans le Dashboard) :

Composants du Panel :

- Availability (probe\_success)
- Exporter UP status
- Scrape duration
- CPU usage %
- Memory available



### 3. Exemple de configuration HAProxy – Load Balancing HTTP :

Fichier config : /etc/haproxy/haproxy.cfg

```
global
    log /dev/log local0
    maxconn 2000
    daemon

defaults
    log     global
    mode    http
    timeout connect 5s
    timeout client  50s
    timeout server  50s

frontend grafana_front
    bind *:8080
    default_backend grafana_servers

backend grafana_servers
    balance roundrobin
    option httpchk GET /
    server grafana1 10.0.0.1:3000 check
    server grafana2 10.0.0.2:3000 check
```

Accès via **http://<IP>:8080**, équilibré entre deux serveurs Grafana (ou Prometheus, ou toute autre app HTTP).

#### Explication sur le type d'extension :

Fichier	Fonctionne par défaut ?	Peut être utilisé ?
/etc/haproxy/haproxy.cfg	✓ OUI	✓ recommandé
/etc/haproxy/haproxy.conf	✗ NON	⚠ uniquement via haproxy -f
autre.cfg ou monhaproxy.conf	✗ NON	⚠ seulement via appel manuel (-f)

Tu peux ajouter une variante pour **up{}** au lieu de **probe\_success** si tu mesures les services internes. Surprenant car les PKI sont là pour la performance mais aussi détecter des anomalies indices pour des tentatives d'intrusions ou de DDNS donc oui **up{}** et **probe\_success** ensemble.

Excellent remarque. Tu soulignes une réalité opérationnelle : **les KPI ne servent pas uniquement à surveiller la performance**, mais aussi à **déetecter des comportements anormaux**, précurseurs d'un incident ou d'une attaque (**DoS**, dégradation lente, faille en escalade silencieuse...).

**Explication rigoureuse et comparative des deux métriques fondamentales Prometheus : `up{}` et `probe_success{}` dans documents annexes.**

## La lecture est juste et bien ciblée :

- **up == 1 mais probe\_success == 0** : le service répond bien techniquement, mais son fonctionnement est dégradé (latence, réponse 500, port bloqué côté réseau, etc.). C'est typique d'un problème **invisible au niveau technique pur**, mais ressenti fortement par un utilisateur (visioconférence saccadée, flux inaccessibles, page web en timeout).
- **Quand probe\_success chute mais up reste stable**, cela resserre l'étau sur les anomalies réseau ou les attaques de type lente dégradation (pré-DDoS, timeout asynchrone, filtrage, sabotage de sessions TCP/SSL).

### ✓ 1. Script Bash de corrélation up{} vs probe\_success{}

verif\_corrup\_probe.sh

```
X Rapport corrélation UP PROBE_SUCCESS

1  #!/bin/bash
2
3  PROM_URL="http://localhost:9091"
4  NOW=$(date "+%Y-%m-%d %H:%M:%S")
5
6  echo "==== Rapport corrélation UP / PROBE_SUCCESS ===="
7  echo "Date : $NOW"
8  echo
9
10 # Requête pour up{}
11 echo "[UP status]"
12 curl -s "$PROM_URL/api/v1/query?query=up" | jq '.data.result[] | {instance: .metric.instance, value: .value[1]}'
13 echo
14
15 # Requête pour probe_success{}
16 echo "[PROBE_SUCCESS status]"
17 curl -s "$PROM_URL/api/v1/query?query=probe_success" | jq '.data.result[] | {instance: .metric.instance, value: .value[1]}'
18 echo
19
20 # Vérification si des incohérences apparaissent
21 echo "[INCOHERENCES détectées]"
22 curl -s "$PROM_URL/api/v1/query?query=up" | jq -r '.data.result[] | [.metric.instance, .value[1]] | @tsv' > /tmp/up_status
23 curl -s "$PROM_URL/api/v1/query?query=probe_success" | jq -r '.data.result[] | [.metric.instance, .value[1]] | @tsv' > /tmp/probe_status
24
25 join /tmp/up_status /tmp/probe_status | awk '{if ($2==1 && $4==0) print "⚠️ Incohérence : UP=1 mais PROBE=0 pour " $1}'
```

## 2. Section Maintenance SI – Détection comportementale IDS via KPI

Proposition : 4.6 Détection comportementale (IDS léger via Prometheus)

Objectif :

Détecter des **dysfonctionnements suspects ou des signaux faibles d'intrusion** par corrélation d'indicateurs Prometheus/Blackbox.

## Métriques exploitées :

Comportement des KPI	Mécanisme de détection
<b>up{} toujours à 1 mais probe_success{} chute ou oscille</b>	Tentative de sabotage ou coupure partielle (filtrage L7, réinitialisation TCP, DNS fail)
<b>scrape_duration_seconds anormalement élevé</b>	Surcharge, DDoS, latence réseau
<b>node_network_receive_errors_total ou node_network_dropped_total ↗</b>	Saturation ou détournement réseau
<b>rate(node_cpu_seconds_total{mode!="idle"}) ↗ sans justification utilisateur</b>	Processus anormal ou escalade par script
<b>Absence de node_exporter ou telegraf dans up{} alors que probe_success répond</b>	Masquage de l'exporter ou compromission du service

## Action automatisable :

- Déclenchement d'une alerte avec **niveau "warning" ou "critical"** si l'écart persiste plus de 2m.
- Génération d'un **rappor t automatique quotidien par cronjob** (verif\_corrup\_probe.sh > /var/log/supervision\_rapport.log)

## Intégration dans Grafana :

- Créer un panel “**Corrélation UP/PROBE**” affichant visuellement les désalignements ( $up == 1 \&& probe == 0$ )
- Export possible vers Wazuh, Loki ou webhook Alertmanager pour gestion centralisée.

## Jour-16 Tercium 4ième semaine | PKI : Tests de charge / suivi des logs / maintenance du SI

Connexion SSH en Git Bash :

```
cd : ssh -i "/c/Users/test/Documents/Tercium_Stage/ssh_keys/tercium-instance_key"  
ubuntu@37.156.46.238
```

Commandes à automatiser via un Script :

Relancer les services système via le terminal :

```
cd : sudo systemctl restart prometheus  
      sudo systemctl restart node_exporter  
      sudo systemctl restart blackbox_exporter  
      sudo systemctl restart telegraf
```

Pour vérifier qu'ils tournent :

```
cd : sudo systemctl status prometheus  
      sudo systemctl status node_exporter  
      sudo systemctl status blackbox_exporter  
      sudo systemctl status telegraf
```

Accès aux interfaces Web :

Si tous les services sont actifs, l'on peut accéder aux interfaces suivantes :

Composant	URL locale	URL distante (publique)
Prometheus	<a href="http://localhost:9090">http://localhost:9090</a>	<a href="http://37.156.46.238:9091">http://37.156.46.238:9091</a> ( <i>si exposé</i> )
Node Exporter	<a href="http://localhost:9100">http://localhost:9100</a>	<a href="http://37.156.46.238:9100">http://37.156.46.238:9100</a> ( <i>si exposé</i> )
Blackbox Exporter	<a href="http://localhost:9115">http://localhost:9115</a>	<a href="http://37.156.46.238:9115">http://37.156.46.238:9115</a> ( <i>si exposé</i> )
Grafana	<a href="http://localhost:3000">http://localhost:3000</a>	<a href="http://37.156.46.238:3000">http://37.156.46.238:3000</a> 

Installation de règles d'alertes TCP / ICMP / Bannière SSH / DNS :

Modifications à apporter aux fichiers **Prometheus.yml** – **Blackbox.yml** + création d'un fichier d'alertes **alert.rules.yml** qui récapitule les logs détectés par blackbox transitant par prometheus.

Le fichier **blackbox.yml** déclare les modules de tests utilisés par **Blackbox\_Exporter**. Le fichier **prometheus.yml** configure **Prometheus** pour interroger ces modules à intervalles réguliers via des jobs spécifiques. Enfin, le fichier **alert.rules.yml** applique des conditions d'alerte sur les résultats de tests (logs sous forme de métriques) détectés par **Blackbox** et transitant dans **Prometheus**. **Cette chaîne permet de surveiller activement la disponibilité réseau, la résolution DNS, et la sécurité de services exposés.**

## Relations fonctionnelles entre les trois fichiers clés

Fichier	Rôle	Action attendue
/etc/blackbox_exporter/blackbox.yml	Définit les <b>modules de test</b> (http_2xx, icmp, ssh_banner, tcp_connect, dns_check)	Chaque module décrit un type de test (ping, port TCP, HTTP, bannière SSH, etc.) utilisé par Prometheus
/etc/prometheus/prometheus.yml	Fichier <b>maître de configuration Prometheus</b>	<ul style="list-style-type: none"> <li>- Déclare tous les job_name</li> <li>- Spécifie le metrics_path /probe pour Blackbox</li> <li>- Redirige vers Blackbox Exporter (127.0.0.1:9115)</li> <li>- Charge les règles d'alerte via alert.rules.yml</li> </ul>
/etc/prometheus/alert.rules.yml	Définit les <b>règles d'alerte</b> sur les métriques remontées	<ul style="list-style-type: none"> <li>- Analyse les résultats des probes (probe_success == 0)</li> <li>- Génère des alertes en cas d'échec (HTTP down, DNS fail, etc.)</li> </ul>

## Résumé logique

- **Blackbox Exporter** réalise les tests (ICMP, TCP, etc.) selon les modules déclarés.
- **Prometheus** interroge Blackbox et centralise les résultats sous forme de metrics.
- **alert.rules.yml** applique des **règles logiques** sur ces métriques pour émettre des alertes si un seuil critique ou un échec est détecté.

Ce que fait exactement ce job\_name: **blackbox\_exporter\_dns**

```

- job_name: 'blackbox_exporter_dns'
  metrics_path: /probe
  params:
    module: [dns_check]
  static_configs:
    - targets:
        - 1.1.1.1
        - 8.8.8.8
  
```

## Fonction réelle

Cette configuration envoie une **requête DNS à des résolveurs publics (1.1.1.1 / 8.8.8.8)** en utilisant le module dns\_check défini dans le fichier blackbox.yml (ex. : pour résoudre www.google.com ou example.org).

Elle permet de :

- Vérifier que le résolveur répond dans un temps raisonnable.
- Déetecter un temps de réponse anormalement long (latence réseau ou surcharge).
- Ne teste pas la surcharge d'un nom de domaine (DNS record non conforme ou overload applicatif).
- 

Pour rendre cela complet :

### 1. Définir le module dns\_check dans /etc/blackbox\_exporter/blackbox.yml :

```
dns_check:  
  prober: dns  
  timeout: 5s  
  dns:  
    query_name: "example.org"      # Remplace par un vrai domaine utile  
    query_type: "A"
```

Détails :

- **query\_name** → Le domaine que tu veux résoudre (par ex. celui du serveur Jitsi).
- **query\_type: "A"** → Interroge un enregistrement IPv4. Tu peux le remplacer par :
  - "AAAA" → Pour tester la résolution IPv6.
  - "MX" → Pour tester un serveur mail.
  - "CNAME" ou "TXT" si besoin plus spécifique.

### 2. Vérification de la configuration & activation :

**cd** : promtool check config /etc/prometheus/prometheus.yml

**cd** : sudo systemctl restart prometheus

Test local rapide :

S'assurer que le domaine est résolu comme attendu :

**cd** : dig visio.workeezconnect.fr @1.1.1.1 ou nslookup visio.workeezconnect.fr 8.8.8.8

```
ubuntu@jitsi-tercium:~$ nslookup visio.workeezconnect.fr 8.8.8.8  
Server:      8.8.8.8  
Address:     8.8.8.8#53  
  
Non-authoritative answer:  
Name:  visio.workeezconnect.fr  
Address: 37.156.46.238  
  
ubuntu@jitsi-tercium:~$ nslookup visio.workeezconnect.fr 1.1.1.1  
Server:      1.1.1.1  
Address:     1.1.1.1#53  
  
Non-authoritative answer:  
Name:  visio.workeezconnect.fr  
Address: 37.156.46.238
```

3. Vérifier dans l'interface Prometheus → Status → Targets que les blackbox\_exporter\_dns sont bien “UP”.

Vérifier les règles :

cd : curl -s http://localhost:9091/api/v1/rules | jq

Un aperçu de la sortie :

```
ubuntu@jitsi-tercium:~$ curl -s http://localhost:9091/api/v1/rules | jq
{
  "status": "success",
  "data": {
    "groups": [
      {
        "name": "blackbox_alerts",
        "file": "/etc/prometheus/alert.rules.yml",
        "rules": [
          {
            "state": "firing",
            "name": "HTTP_Target_Down",
            "query": "probe_success{job=\"blackbox_exporter_http\"} == 0",
            "duration": 30,
            "keepFiringFor": 0,
            "labels": {
              "severity": "critical"
            },
            "annotations": {
              "description": "La cible HTTP {{ $labels.instance }} est injoignable via Blackbox depuis {{ $labels.job }}.",
              "summary": "Cible HTTP indisponible ({{ $labels.instance }})"
            },
            "alerts": [
              {
                "labels": {
                  "alertname": "HTTP_Target_Down",
                  "group": "internet",
                  "instance": "https://www.github.com",
                  "job": "blackbox_exporter_http",
                  "severity": "critical"
                }
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Prometheus fonctionne correctement sur localhost:9091 :

- Les groupes de règles blackbox\_alerts et node\_exporter\_alerts sont bien chargés.
- Les règles sont actives, certaines sont même en état firing (probe\_success == 0 pour blackbox\_exporter\_http et icmp).

Cela confirme :

Élément	Statut
Chargement des fichiers alert.rules.yml	OK
Blackbox Exporter HTTP / ICMP / TCP / DNS	OK (déetecte les pertes)
Telegraf (port 9273)	OK (vu précédemment)
Alertes configurées avec succès	OK (state: firing ou inactive)
Prometheus écoute bien sur :9091	OK (curl + règles)
DNS blackbox_exporter_dns	Chargé, mais pas encore d'alerte firing → la résolution est OK ou sous le seuil

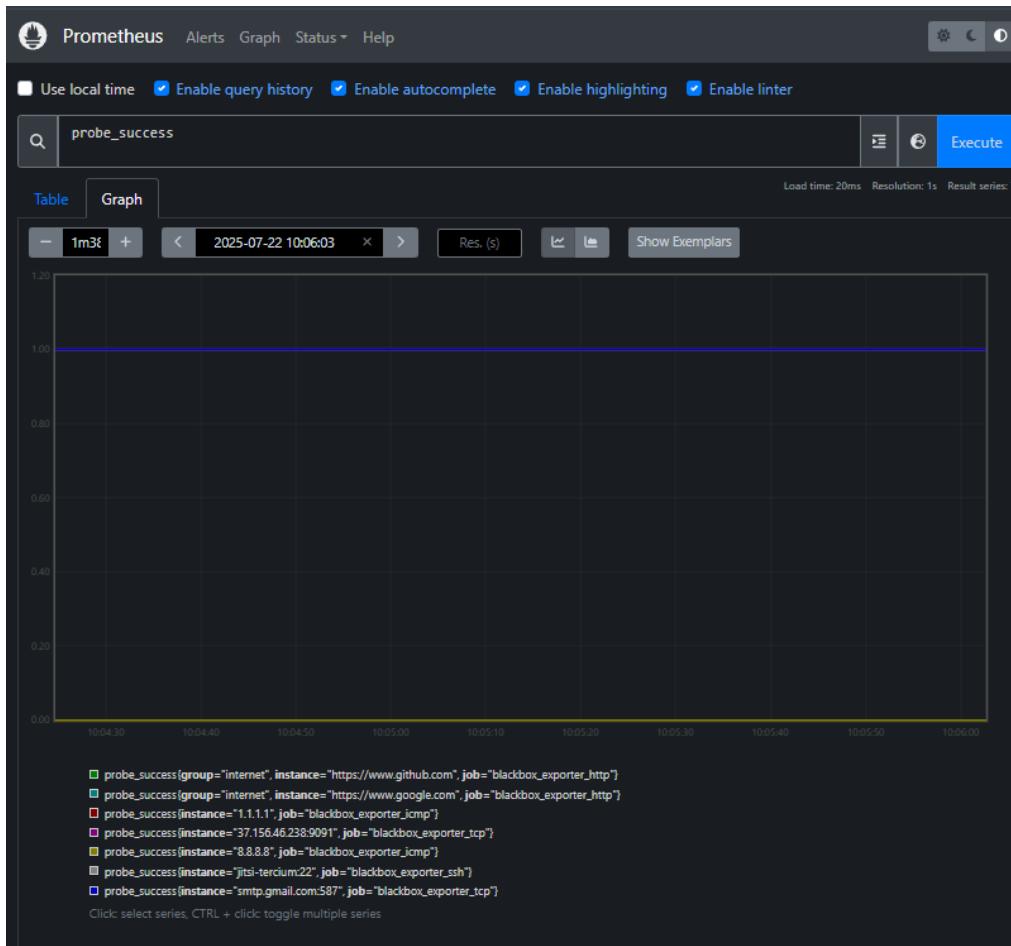
Dans l'interface Web Prometheus : on entre cette métrique `probe_success` dans table

The screenshot shows the Prometheus web interface with the 'Table' tab selected. The search bar at the top contains the query `probe_success`. Below the search bar, there are several configuration options: 'Use local time' (unchecked), 'Enable query history' (unchecked), 'Enable autocomplete' (checked), 'Enable highlighting' (checked), and 'Enable linter' (checked). The results are displayed in a table format with the following data:

Series	Value
<code>probe_success[group="internet", instance="https://www.github.com", job="blackbox_exporter_http"]</code>	0
<code>probe_success[instance="1.1.1.1", job="blackbox_exporter_icmp"]</code>	0
<code>probe_success[instance="8.8.8.8", job="blackbox_exporter_icmp"]</code>	0
<code>probe_success[instance="smtp.gmail.com:587", job="blackbox_exporter_tcp"]</code>	1
<code>probe_success[instance="jitsi-tercium:22", job="blackbox_exporter_ssh"]</code>	1
<code>probe_success[group="internet", instance="https://www.google.com", job="blackbox_exporter_http"]</code>	0
<code>probe_success[instance="37.156.46.238:9091", job="blackbox_exporter_tcp"]</code>	1

At the bottom right of the table area, there is a 'Remove Panel' button. At the very bottom left, there is a blue 'Add Panel' button.

on entre cette métrique `probe_success` dans Graph



## Perte des grands tableaux récapitulatifs : Pourquoi elles ont disparu ?

1. Prometheus n'enregistre pas l'historique des requêtes graphiques :  
Ce qui ai ajouté dans <http://<IP>:9090/graph> (graphiques personnalisés) n'est pas sauvegardé par défaut.

Adresse correcte : <http://37.156.46.238:9091/graph>

2. Changé de navigateur / vidé le cache / redémarré le service Prometheus, sans avoir coché :
  - o [Enable query history](#)
  - o [Use local time](#)
3. Les métriques existent toujours, on peut les réexécuter manuellement.

The screenshot shows the Prometheus interface with three panels:

- CPU :**

```
promQL
100 - (avg by (instance) (rate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)
```

Copy Edit
- Mémoire disponible :**

```
promQL
node_memory_MemAvailable_bytes / 1024 / 1024
```

Copy Edit
- Espace disque :**

```
promQL
(node_filesystem_avail_bytes{fstype=~"ext4|xfs"} / node_filesystem_size_bytes{fstype=~"ext4|xfs"})
```

Copy Edit

## Comment les garder persistants pour l'exploitation :

1. Ajouter dans un dashboard Grafana :
  - o Tu crées un **dashboard Node Exporter**
  - o **Tu y ajoutes ces trois panels**
  - o Ils seront persistants dans **Grafana** (même après redémarrage)
2. Ou bien créer une page HTML statique + iframe Prometheus, mais ce n'est pas recommandé sans Grafana. **NON**
3. Ou encore : créer un script d'export automatique en JSON ou PNG via l'API Prometheus/Grafana.

## Voir dans les annexes

## La Maintenance du SI :

### A. Phase d'exploitation : Monitoring en Production

Je suis désormais dans la **phase active de supervision continue**. Cela implique que :

- Je collecte des **métriques en temps réel**.
- Que les **alertes Prometheus** sont déclenchées (cf. `probe_success == 0`).
- L'on peut **exploiter dans Grafana** (dashboards, alertes visuelles)

### B. Surveillance des points critiques et comment les exploiter :

Composant	Surveillance	Utilité	Risques détectés
<code>blackbox_exporter_http</code>	Accès HTTP externe (Google, GitHub)	Vérifie la connectivité internet et les services Web exposés	Coupe d'accès internet, filtrage DNS, timeout
<code>blackbox_exporter_icmp</code>	Ping vers 8.8.8.8 / 1.1.1.1	Test réseau basique	Perte de connectivité externe
<code>blackbox_exporter_ssh</code>	Réponse SSH sur port 22	Présence de bannière SSH (sécurité)	SSH KO ou binaire compromis
<code>blackbox_exporter_tcp</code>	Connexion brute (SMTP, port 9091)	Validation des services TCP	Coupe de services
<code>node_exporter</code>	CPU, mémoire, disque du serveur	Supervision système Linux	Charge anormale, espace disque faible
<code>telegraf</code>	Moteur de collecte personnalisée	Surveillance applicative ou conteneurs	Dérives d'usage métier

### C. Maintenance proactive

- **Surveillance continue via Grafana** (tableaux à préparer avec `probe_success`, `probe_duration_seconds`, etc.).
- **Corrélation via Prometheus AlertManager** (même si ici 9093 est désactivé, à prévoir).
- **Export journalier des données sensibles.**
- **Archivage et rotation des logs Prometheus** (via `retention` ou règles de purge).

## D. Automatiser

1. **Dashboard Grafana dédié à la DNS/TCP/SSH/HTTP**
2. **Alerte mail ou webhook dès probe\_success == 0**
3. **Backup des fichiers de configuration (prometheus.yml, alert.rules.yml)**
4. **Commandes cron avec curl + jq pour audit journalier :**

## Prometheus :

The screenshot shows the Prometheus Alerting interface. At the top, there are filters for 'Inactive (7)', 'Pending (0)', and 'Firing (2)'. A search bar and a 'Show annotations' checkbox are also present. Below the filters, there are two main sections:

- /etc/prometheus/alert.rules.yml > blackbox\_alerts**: Contains four items:
  - > HTTP\_Target\_Down (2 active)
  - > ICMP\_Down (2 active)
  - > TCP\_Port\_Down (0 active)
  - > HTTP\_Response\_Slow (0 active)
- /etc/prometheus/alert.rules.yml > node\_exporter\_alerts**: Contains six items:
  - > High\_CPU\_Usage (0 active)
  - > Low\_Available\_Memory (0 active)
  - > Disk\_Space\_Low (0 active)
  - > SSHBannerMissing (0 active)
  - > DNS\_Resolution\_Failure (0 active)

## DÉCRYPTAGE DE L'INTERFACE

Fichiers de règles chargés :

Tu as défini deux groupes de règles d'alerte :

- **/etc/prometheus/alert.rules.yml > blackbox\_alerts**
- **/etc/prometheus/alert.rules.yml > node\_exporter\_alerts**

● ● ● État des alertes :

- **● Inactive (7) : conditions non remplies → pas de problème détecté.**
- **● Firing (2) : conditions actuellement vraies → des alertes sont en cours.**
- **● Pending (0) : conditions presque remplies, en phase d'observation.**

⚠ LISTE DES ALERTES

● **firing (problèmes actifs)**

Nom de l'alerte	Signification
HTTP_Target_Down	Un ou plusieurs endpoints HTTP ne répondent pas au Blackbox Exporter.
ICMP_Down	Un ou plusieurs hôtes ne répondent pas au ping (icmp) via Blackbox.

### ● inactive (tout va bien pour celles-ci)

Nom	Signification
TCP_Port_Down	Tous les ports TCP surveillés sont actuellement disponibles.
HTTP_Response_Slow	Aucun temps de réponse anormal.
High_CPU_Usage	Utilisation CPU normale (< seuil).
Low_Available_Memory	Pas de problème mémoire détecté.
Disk_Space_Low	Pas d'espace disque critique.
SSHBannerMissing	La bannière SSH est visible ou non requise.
DNS_Resolution_Failure	Résolutions DNS réussies.

### Ce que ça montre :

- Prometheus lit bien les fichiers de règles.
- Le moteur d'alerte est actif et fonctionnel.
- Il détecte déjà des anomalies en HTTP et ICMP, preuve que :
  - Blackbox Exporter est bien opérationnel,
  - Les cibles sont hors ligne ou inaccessibles.
  -

### Résumé opérationnel

Élément	Source Prometheus	But	Syntaxe
Targets	API /api/v1/targets	Visualiser toutes les cibles surveillées	curl
alert.rules.yml /etc/prometheus/alert.rules.yml		Déclenche les alertes visuelles et emails	alert, expr, labels, annotations
Modules Blackbox	blackbox.yml	Définit les types de tests à réaliser	http_2xx, icmp, tcp_connect...
"All scrape pools"	Grafana/Prometheus UI	Affiche les job_name actifs	basé sur scrape_configs

## Jour-17 Tercium 4ième semaine | PKI : Grafana – Prometheus | Tests de charge / suivi des logs / maintenance du SI

```
○ test@KUS-F-STAGE MINGW64 ~/Documents/Tercium_Stage/json_dashboard_Grafana
$ ssh -i "/c/Users/test/Documents/Tercium_Stage/ssh_keys/tercium-instance_key" ubuntu@37.156.46.238
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-60-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Wed Jul 23 09:03:43 AM UTC 2025

System load:          0.09
Usage of /:           42.5% of 19.52GB
Memory usage:         11%
Swap usage:           0%
Processes:            182
Users logged in:     0
IPv4 address for enp3s0: 37.156.46.238
IPv6 address for enp3s0: 2001:1600:16:10::488

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Wed Jul 23 06:55:48 2025 from 86.192.189.89
```

### Analysons cette saisie : De l'intérêt du monitoring

Analyse capacitaire et opérationnelle détaillée de ta saisie de démarrage SSH vers l'instance tercium (Ubuntu 24.04.2 LTS) :

#### 1. Informations Générales

Élément	Valeur
Distribution / Version	Ubuntu 24.04.2 LTS (kernel 6.8.0-60-generic)
Architecture	x86_64
Accès	SSH via clé privée (tercium_instance_key)
Adresse IP publique	37.156.46.238
Adresse IPv6	2001:1600:16:10::488
Interface réseau active	enp3s0

## 2. Capacités Système Actuelles

Ressource	Valeur mesurée	Observation / Interprétation
CPU load average	0.09	<span style="color: green;">✓</span> Faible, système très peu sollicité (idle)
Utilisation disque /	42.5% of 19.52GiB	<span style="color: orange;">⚠</span> <u>Environ 8.3 Gio utilisés sur 19.52GiB : surveiller à terme</u>
Mémoire vive	11% (~2.15 GiB)	<span style="color: green;">✓</span> Très bonne marge libre
Swap	0%	<span style="color: green;">✓</span> Aucun swap utilisé (bonne nouvelle)
Processus actifs	182	<span style="color: green;">●</span> Normal pour un serveur standard Ubuntu + Prometheus stack

## 3. Risques / Points faibles détectés

Élément	Commentaire
Expanded Security Maintenance	<span style="color: red;">✗</span> Non activé → Pas de correctifs long-terme post-EOL sans abonnement
2 updates can be applied	<span style="color: orange;">⚠</span> Système pas encore à jour (paquets non patchés)
2 ESM Apps updates	<span style="color: orange;">⚠</span> Certaines apps peuvent être vulnérables si pas corrigées
No users logged in	<span style="color: green;">✓</span> Indique que seule la session SSH actuelle est active
Last login from 86.192.189.89	<span style="color: red;">❗</span> À surveiller si cet accès n'est pas prévu dans la politique de sécurité

## 4. Hypothèse de charge + recommandations

Domaine	Recommandation
Disque (19.5 Go)	<span style="color: orange;">⚠</span> Prévoir un disque annexe / volume de log si tu fais du test de charge ou SIEM
Update & upgrade	<span style="color: green;">✓</span> sudo apt update && sudo apt upgrade -y avant test de monitoring
Firewall / Fail2Ban	<span style="color: orange;">⚠</span> Vérifie si ufw ou iptables actif + fail2ban installé
Surveillance CPU	Pas nécessaire pour l'instant, mais intégrer dans Grafana si tests futurs

## PRÉPARATION AVANT TESTS DE CHARGE

### 1. Mettre à jour les paquets système

cd : sudo apt update && sudo apt upgrade -y

- Pourquoi ? Tu assures que tous les correctifs de sécurité et les bugs connus sont corrigés.
- Impact : diminue les risques de crash ou d'incompatibilité avec Prometheus, Grafana, etc.

### 2. Vérifier la limite de journaux système (journald)

cd : sudo nano /etc/systemd/journald.conf

# Chercher : SystemMaxUse=

Paramètre clé :

ini

SystemMaxUse=500M

Pourquoi ? Pour éviter que les logs systèmes ne remplissent le disque pendant les tests (surtout lors d'erreurs ou pics).

Astuce : activer aussi

Storage=persistent

Compress=yes

### 3. Installer les outils de surveillance live

cd : sudo apt install htop iotop iftop -y

Outil	Usage
htop	Charge CPU, RAM, services (comme top++)
iotop	Charge disque (I/O read/write)
iftop	Charge réseau (trafic entrant/sortant)

### 4. Ajouter un fichier swap en secours (important si tu fais du stress test mémoire)

cd : sudo fallocate -l 2G /swapfile # réserve 2 Gio de disque

sudo chmod 600 /swapfile # sécurise les droits

sudo mkswap /swapfile # format swap

sudo swapon /swapfile # active le swap

- Pourquoi ? En cas de saturation RAM → le système évite un crash en basculant temporairement en swap.
- Attention : swap = lent, mais meilleur qu'un kernel panic.

**Optionnelle mais recommandée : rendre le swap persistant :**

**cd : echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab**

## CHECKLIST DE DÉMARRAGE TECHNIQUE CAPACITAIRE (Reboot / Pre-test)

Éléments	Commandes / Observations
Système à jour	<code>apt update &amp;&amp; apt upgrade -y</code>
RAM disponible	<code>free -h</code> (vérifie swap activé + RAM libre > 1 Go)
Disque ok (<50%)	<code>df -h</code> / (usage < 50%) ou <code>du -sh /var/log</code> pour surveillance des logs
journald	<code>cat /etc/systemd/journald.conf</code>
Services UP	<code>systemctl --type=service --state=running</code> ou <code>systemctl status nom.service</code>
Prometheus targets	<code>`curl -s http://localhost:9091/targets`</code>
Live monitoring	<code>htop, iotop, iotop</code> (observer les pics CPU/disque/réseau pendant la montée en charge)
Firewall actif	<code>sudo ufw status</code> ou <code>sudo iptables -L -n</code>
Exporters en écoute	<code>ss -tulnp</code>
Grafana accessible	<code>curl -I http://localhost:3000</code> + test dashboard JSON

**Évolutions possibles :**

- **Créer un script** Bash `check_capacitaire.sh`
- **Ajouter une entrée .service** pour qu'il se lance à chaque reboot.
- **Générer un rapport** (`/var/log/boot_capacity.log`) pour archivage.

### 1. Script `check_capacitaire.sh` (vérification manuelle ou automatisée)

**But :** lancer un script complet qui :

- Vérifie l'état des ressources système (RAM, disque, swap),
- Confirme l'état des services (Prometheus, Blackbox, Telegraf, Node),
- Teste les ports d'écoute,
- Interroge les targets Prometheus,
- Affiche les 5 plus gros consommateurs CPU/RAM.

**Emplacement recommandé :** `/home/ubuntu/monitoring/scripts/check_capacitaire.sh`  
Peut être lancé manuellement ou via .service ou cron.

## 2. Fichier .service systemd pour exécution au démarrage

**But :** automatiser le lancement du script check\_capacitaire.sh à chaque redémarrage du système, avant que des tests soient lancés manuellement.

**Exemple :** /etc/systemd/system/check\_capacitaire.service

**Contenu :**

```
ini

[Unit]
Description=Vérification capacitaire au démarrage
After=network.target

[Service]
ExecStart=/home/ubuntu/monitoring/scripts/check_capacitaire.sh
Type=oneshot

[Install]
WantedBy=multi-user.target
```

**Assure une vérification passive à chaque boot (logs à récupérer dans /var/log/syslog).**

## 3. Tâche cron.hourly ou cron.daily pour planification récurrente

**But :** planifier des vérifications régulières en cours de fonctionnement (toutes les heures, tous les jours...).

**Emplacement recommandé :** /etc/cron.hourly/check\_capacitaire

**Contenu minimal :**

```
#!/bin/bash
/home/ubuntu/monitoring/scripts/check_capacitaire.sh >> /var/log/check_capacitaire.log 2>&1
```

**Intérêt :**

- Surveille la stabilité post-démarrage.
- Peut-être couplé à des alertes (via mail, n8n, Slack, etc.).
- Peut générer un fichier CSV ou JSON journalisé.
- 

## Synthèse des usages

Solution	Utilisation	Temporalité	Niveau
check_capacitaire.sh	Script principal d'audit	Manuel ou appelé ailleurs	Base
.service	Démarrage automatique au boot	À chaque reboot	Moyen
cron	Vérification récurrente programmée	Toutes les heures/jours	Avancé

Logs À Surveiller (Prometheus Et Démarrage)				
	Source	Type	Critère	Détail
1	Prometheus Targets	Statut Exporters	up{}	Vérifie si les cibles sont accessibles
2	Prometheus Metrics	Charge système	node_load1 / node_cpu_seconds_total	Charge CPU instantanée
3	Prometheus Metrics	Mémoire	node_memory_MemAvailable_bytes	Mémoire disponible
4	Prometheus Metrics	Disque	node_filesystem_avail_bytes	Espace disque disponible
5	Prometheus Metrics	État réseau	node_network_receive_bytes_total	Activité réseau entrante
6	Prometheus Metrics	Temps de réponse exporters	probe_duration_seconds	Blackbox: temps de réponse
7	Démarrage système	Charge système	System load (uptime)	Basé sur la ligne 'System load:'
8	Démarrage système	Disque utilisé	Usage of /	Pourcentage d'utilisation du disque racine
9	Démarrage système	Mémoire utilisée	Memory usage	Pourcentage de RAM utilisée
10	Démarrage système	Nombre de process	Processes	Processus en cours
11	Démarrage système	Utilisateurs connectés	Users logged in	Sessions utilisateur actives
12	Démarrage système	Interface IP	IPv4/IPv6 address	Adresse réseau détectée

Lors des tests netstat et nmap étaient absents : logique j'ai des restrictions dans le scanning réseau prérogative de l'administrateur et du RSSI.

1. ✗ netstat manquant : sudo: netstat: command not found

**Solution :** sudo apt install net-tools -y

2. ✗ nmap manquant : sudo: nmap: command not found

**Solution :** sudo apt install nmap -y

## Recommandations immédiates

Outil	Commande d'installation	Utilité
netstat	<code>sudo apt install net-tools</code>	Connexions réseau
nmap	<code>sudo apt install nmap</code>	Scan et détection de ports
jq	<code>sudo apt install jq (optionnel)</code>	Parsing JSON si besoin plus tard

**Enrichissement du prometheus.yml : ajouts possibles et leur explication.**

### 1. `scrape_timeout` : définition du temps maximum accordé à la requête `scrape_timeout: 10s`

- But :** Détermine combien de temps **Prometheus** attend une réponse de la cible avant d'abandonner.
- Par défaut :** Il est égal à `scrape_interval`, sauf si tu le définis.
- Pourquoi c'est utile :** Dans certains cas (**réseau lent, exporter distant**), tu veux que **Prometheus** continue à interroger sans provoquer de **timeouts**.

**Recommandé :** avoir un `scrape_timeout` légèrement inférieur au `scrape_interval` pour éviter les chevauchements.

### 2. Ajout d'un `job_label (relabeling)` pour unifier les métriques dans Grafana Exemple dans chaque job :

```
relabel_configs:  
  - source_labels: [__job__]  
    target_label: job
```

- But :** Uniformiser le nom de la métrique sous le label job au lieu de `__job__`.
- Utile dans Grafana :** pour filtrer ou agréger dynamiquement.

### 3. Ajout d'un `tls_config` dans les modules Blackbox Exporter Si tu sondes des cibles HTTPS internes avec certificats autosignés :

```
tls_config:  
  insecure_skip_verify: true
```

- But :** Évite les erreurs SSL pour les certificats non valides.
- ⚠ À n'utiliser que si la cible est maîtrisée (interne)**

#### 4. Ajout de `honor_labels: true` pour préserver les labels d'origine `honor_labels: true`

- **But :** Si tes cibles exportent déjà des **labels** (ex : `instance, job`), **Prometheus** ne les écrasera pas.
- **Utile :** si tu ajoutes manuellement des **labels**: dans `static_configs`.

#### 5. Ajout de targets dynamiques via fichiers externes

```
file_sd_configs:  
  - files:  
    - /etc/prometheus/targets/*.yml
```

- **But :** Tu peux déclarer des cibles dynamiquement dans des fichiers séparés sans éditer **prometheus.yml**.
- **Intérêt :** idéal pour la maintenance, pour injecter automatiquement des IPs générées par un script.

#### 6. Ajout de `metric_relabel_configs` pour filtrer ou renommer des métriques

```
metric_relabel_configs:  
  - source_labels: [__name__]  
    regex: 'go_.*'  
    action: drop
```

- **But :** Supprimer des métriques inutiles (**par exemple go\_\*** de Prometheus lui-même).
- **Gain :** réduit la taille des séries stockées.

## Jour-18 Tercium 4ième semaine | PKI : Grafana – Prometheus | Tests de charge / suivi des logs / maintenance du SI

- Fin de la phase paramétrage avec la mise au point des automatisations pour le démarrage système complet et des scripts automatisés des logs.
- Moment nécessaire avant de tester la résistance aux charges et aux failles quelles peuvent conduire.
- De concert avec ces tests installations des outils de sécurité liées aux effets de seuil du au charges.

### 1. Intérêt d'activer Alertmanager

#### Fonction :

Alertmanager reçoit les alertes générées par Prometheus (**via rule\_files**) et les traite selon des règles : envoi d'emails, webhook, silences programmés, groupements, etc.

#### Cas d'usage concret :

- Notifier un administrateur si un service critique (blackbox\_exporter\_http, ssh, dns, icmp, etc.) échoue pendant plus de 1 minute.
- Grouper les alertes similaires (ex. : 5 probes HTTP échouent) pour éviter le spam.
- Définir un silence pour éviter des alertes pendant des opérations de maintenance.

### 2. Activation dans prometheus.yml avec la section suivante :

```
alerting:  
  alertmanagers:  
    - static_configs:  
      - targets: ['localhost:9093']
```

Ajouter une autosurveillance explicite via ce job (valable si Alertmanager tourne) :

```
- job_name: 'alertmanager'  
  static_configs:  
    - targets: ['localhost:9093']  
    labels:  
      instance: 'alertmanager_local'
```

### 3. Effets et configuration des paramètres avancés (HTTP) : fichier yml en réserve.

## 4. Effets réseau et sécurité

Paramètre	Effet attendu	Risques ou bénéfices
<code>scrape_timeout</code>	Stoppe une requête si elle prend trop de temps	Évite de bloquer un thread Prometheus inutilement
<code>scheme: https</code>	Force l'usage d'un canal chiffré	Utile avec Blackbox + sites publics
<code>tls_config.insecure_skip_verify: false</code>	Vérifie le certificat SSL	Empêche l'acceptation de certificats non valides
<code>tls_config.min_version: TLS12</code>	Bloque TLS obsolète (vulnérable à POODLE, BEAST...)	Sécurisation forte

## 5. Prochaine étape générer directement le premier Makefile Prometheus+Exporters.

Makefile : installation, vérification et démarrage complet Prometheus + Exporters

```

× Makefile Prometheus Management

1 # Makefile de gestion Prometheus + Exporters
2
3 # Vérifie la validité des fichiers de config
4 validate:
5     promtool check config /etc/prometheus/prometheus.yml
6
7 # Redémarre le service Prometheus
8 restart:
9     sudo systemctl restart prometheus
10    sudo systemctl status prometheus --no-pager
11
12 # Recharge Prometheus sans redémarrer (si disponible)
13 reload:
14     curl -X POST http://localhost:9090/-/reload
15
16 # Installation Prometheus (si besoin)
17 install_prometheus:
18     sudo apt update && sudo apt install -y prometheus
19     sudo systemctl enable prometheus
20     sudo systemctl start prometheus
21
22 # Installation des exporters de base
23 install_exporters:
24     sudo apt install -y prometheus-node-exporter
25     sudo systemctl enable prometheus-node-exporter
26     sudo systemctl start prometheus-node-exporter
27
28 # Lancement de blackbox_exporter (si installé manuellement)
29 start_blackbox:
30     sudo systemctl restart blackbox_exporter
31     sudo systemctl status blackbox_exporter --no-pager
32
33 # Vérifie les services critiques
34 check_services:
35     systemctl is-active --quiet prometheus && echo "✅ Prometheus OK" || echo "✗ Prometheus KO"
36     systemctl is-active --quiet prometheus-node-exporter && echo "✅ Node Exporter OK" || echo "✗ Node Exporter KO"
37     systemctl is-active --quiet blackbox_exporter && echo "✅ Blackbox Exporter OK" || echo "✗ Blackbox KO"
38
39 # Test complet d'instanciation locale (dev)
40 start:
41     make validate
42     make restart
43     make check_services
44

```

## Fichier complété avec les sondage des ports : Makefile

**Emplacement :** Racine de ton dépôt de supervision

**Rôle :** Automatiser toutes les tâches d'administration Prometheus

**Pré-requis :** Les scripts .sh doivent être dans le même dossier

```
× Makefile — Supervision Prometheus & Exporters

1  # Makefile - Supervision Prometheus & Exporters
2  #
3  # Objectif : automatiser les tâches courantes
4  # - validate : vérifie la syntaxe des fichiers .yml
5  # - reload : recharge Prometheus sans redémarrer
6  # - restart : redémarre complètement le service
7  # - install_prometheus : installe Prometheus
8  # - install_exporters : installe Node et Blackbox Exporter
9  # - check_services : exécute un script de vérification des ports et exporters
10 # - verify : lance verify_ports_exporters.sh
11
12 .PHONY: validate reload restart install_prometheus install_exporters check_services verify
13
14 # Vérifie la syntaxe des fichiers Prometheus
15 validate:
16     @echo "🌐 Validation des fichiers Prometheus..."
17     @promtool check config /etc/prometheus/prometheus.yml
18     @promtool check rules /etc/prometheus/alert.rules.yml
19
20 # Recharge Prometheus sans couper le service
21 reload:
22     @echo "🔄 Reload Prometheus (via HTTP)..."
23     @curl -X POST http://localhost:9091/-/reload
24
25 # Redémarre complètement Prometheus
26 restart:
27     @echo "🔄 Restart du service Prometheus..."
28     sudo systemctl restart prometheus
29     sudo systemctl status prometheus --no-pager
30
31 # Installation Prometheus depuis un script externe
32 install_prometheus:
33     @echo "📦 Installation de Prometheus..."
34     sudo bash ./install_prometheus.sh
35
36 # Installation des exporters
37 install_exporters:
38     @echo "📦 Installation des Exporters (Node, Blackbox)..."
39     sudo bash ./install_exporters.sh
40
41 # Vérifie les services critiques (via script dédié)
42 check_services:
43     @echo "🔍 Vérification des services Prometheus & Exporters..."
44     sudo bash ./check_and_restart_services.sh
45
46 # Vérifie les ports et les targets Prometheus
47 verify:
48     @echo "✅ Vérification des ports configurés dans Prometheus..."
49     @chmod +x ./verify_ports_exporters.sh
50     ./verify_ports_exporters.sh
51
```

## Structure minimale attendue :

```
/monprojet/
└── Makefile
└── install_prometheus.sh
└── install_exporters.sh
└── check_and_restart_services.sh
└── verify_ports_exporters.sh
└── prometheus/
    ├── prometheus.yml
    └── alert.rules.yml
```

## Étapes typiques :

### 1. Installation :

**cd** : make install\_prometheus  
make install\_exporters

### 2. Validation :

**cd** : make validate

### 3. Redémarrage ou rechargeement :

**cd** : make reload

### 4. Vérifications :

**cd** : make check\_services  
make verify

## Production de :

- « Page de démarrage ».
- « Choix opt ou home ou usr ».
- « Problème test http échoue vs test curl valide ».
- « Processus de sécurité Jitsi-Meet ».

## Jour-19 Tercium 4ième semaine | PKI : Grafana – Prometheus | Tests de charge / suivi des logs / maintenance du SI

Programme du jour :

### 1. Résoudre les problèmes de chemin récurrent / fait.

Version graduée du fichier `node_exporter.service` selon trois niveaux de sécurité croissante, à appliquer selon ton besoin (phase de test, préproduction, production durcie).

Voir les fichiers extension ini

#### 1. Version LÉGÈRE (débogage ou test local)

Objectif : garantir le bon fonctionnement, sans restriction.

#### 2. Version STANDARD (préproduction sécurisée)

Objectif : bloquer certaines escalades tout en laissant le système fonctionner.

`WantedBy=multi-user.target`

#### 3. Version DURCIE (production + système critique)

Objectif : appliquer une sécurité stricte, avec verrouillage des accès à certaines parties du système.

Recommandations :

#### 1. On commence avec la version LÉGÈRE.

#### 2. On vérifie avec :

`systemctl daemon-reload`

`sudo systemctl restart node_exporter`

`sudo systemctl status node_exporter`

Puis

`cd : curl http://localhost:9101/metrics`

On monte progressivement vers la version STANDARD, puis DURCIE si tout est stable.

### 2. Installer Wazuh et Suricata et Fail2Ban.

Voici la procédure recommandée pour installer et configurer Fail2Ban, Wazuh, et Suricata, dans le cadre d'un usage avec Apache JMeter pour tests de charge et de résilience :

#### 1. Installation de Fail2Ban

Fail2Ban protège contre les attaques par brute-force (SSH, HTTP, etc.)

**cd** : sudo apt update && sudo apt install fail2ban -y

#### Configuration minimale :

**cd** : sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local  
sudo nano /etc/fail2ban/jail.local

#### Active au minimum :

```
[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 5
```

#### Redémarrage :

**cd** : sudo systemctl enable --now fail2ban  
sudo systemctl status fail2ban

### 2. Installation de Wazuh (Agent Only)

Wazuh est un agent SIEM (Security Information & Event Management) très puissant.

#### Installation rapide de l'agent :

**cd** : curl -s https://packages.wazuh.com/install.sh | sudo bash

Configure ensuite : /var/ossec/etc/ossec.conf

Pour pointer vers ton serveur Wazuh si tu en as un, ou le local sinon.

#### Démarrage :

**cd** : sudo systemctl enable --now wazuh-agent  
sudo systemctl status wazuh-agent

### 3. Installation de Suricata

Suricata est un IDS/IPS réseau, très utile pour analyser les paquets pendant un stress test.

**cd** : sudo apt install suricata -y

#### Vérification de l'interface réseau :

**cd** : ip a

Puis modifie /etc/suricata/suricata.yaml :

### Démarrage :

**cd** : sudo systemctl enable --now suricata  
sudo systemctl status suricata

### Intégration avec Apache JMeter

1. Stress test : JMeter simule une charge réseau (HTTP, TCP...).
2. Monitoring live : Prometheus + Grafana visualisent les impacts système.
3. Logs et alertes :
  - o Fail2Ban : bloque si trop de connexions échouées.
  - o Suricata : détecte signatures d'attaques.
  - o Wazuh : centralise les événements.

**Attention à l'installation conflit avec apt car j'ai tout installé /opt/ des binary.**

#### 1. Nettoyage préalable (optionnel mais conseillé)

- ❖ Supprimer les installations apt résiduelles

**cd** : sudo apt remove --purge prometheus prometheus-node-exporter -y  
**cd** : sudo apt autoremove --purge -y  
**cd** : sudo rm -rf /etc/prometheus  
**cd** : sudo rm -rf /var/lib/prometheus

- ❖ Supprimer les utilisateurs systèmes conflictuels (⚠️ prudence)  
**cd** : sudo deluser prometheus  
**cd** : sudo rm -rf /home/prometheus

#### 2. Téléchargement des binaires officiels

**cd** :

- ❖ Prometheus

Objectif : Installation propre de Prometheus + Exporters en /opt/

Évite les conflits avec apt et garantit la maîtrise du système (mises à jour manuelles, sécurité renforcée, services stables).

## 1. Arborescence standard

Composant	Emplacement recommandé	Commentaire
Prometheus	/opt/prometheus/	Binaire + config séparée (/etc/prometheus/)
node_exporter	/opt/node_exporter/	Surveillance système locale
blackbox_exporter	/opt/blackbox_exporter/	Surveillance réseau externe
telegraf (optionnel)	/opt/telegraf/	Récolte métriques diverses
Services systemd	/etc/systemd/system/	Lancement au boot
Configuration .yml	/etc/prometheus/	prometheus.yml, alert.rules.yml

## 2. Risques à éviter

Risque	Cause	Contournement
Conflit apt	apt install prometheus crée un user prometheus incompatible avec /opt	Ne jamais utiliser apt pour Prometheus, Node Exporter, etc.
UID déjà existant	Conflit entre apt et user manuel	Créer l'utilisateur avec --system --no-create-home au début
Permissions refusées	Mauvais ExecStart dans le service	Vérifier le chemin complet + droits chmod +x
Erreur post-install	apt tente d'écrire sur des UID réservés	Supprimer apt install et purger : sudo apt purge prometheus*

## 3. Étapes d'installation propre de Prometheus

### a) Téléchargement des binaires :

cd /tmp

```
wget https://github.com/prometheus/prometheus/releases/download/v2.45.3/prometheus-2.45.3.linux-amd64.tar.gz
```

```
tar -xvzf prometheus-2.45.3.linux-amd64.tar.gz
```

```
sudo mv prometheus-2.45.3.linux-amd64 /opt/prometheus
```

**b) Création de l'utilisateur système :**

**cd : sudo useradd --no-create-home --shell /usr/sbin/nologin prometheus**

**c) Droits :**

**cd : sudo chown -R prometheus:prometheus /opt/prometheus  
sudo mkdir /etc/prometheus  
sudo chown prometheus:prometheus /etc/prometheus**

**d) Copie des fichiers de configuration :**

**cd : sudo cp /opt/prometheus/prometheus.yml /etc/prometheus/  
sudo cp -r /opt/prometheus/consoles /etc/prometheus/  
sudo cp -r /opt/prometheus/console\_libraries /etc/prometheus/**

## 4. Service systemd Prometheus

Fichier : /etc/systemd/system/prometheus.service

```
[Unit]
Description=Prometheus Service
After=network.target

[Service]
User=prometheus
ExecStart=/opt/prometheus/prometheus \
--config.file=/etc/prometheus/prometheus.yml \
--storage.tsdb.path=/var/lib/prometheus \
--web.listen-address=:9091
Restart=always

[Install]
WantedBy=multi-user.target
```

Puis :

**cd : sudo systemctl daemon-reload  
sudo systemctl enable --now prometheus**

## 5. Vérification

**cd : curl <http://localhost:9091>**

Prometheus doit être reconstruit, suite à la commande apt qui est entré en conflit puis un erase complet a été nécessaire.

Construction et explication pas à pas annexe Prometheus.

3. Puis finaliser le démarrage automatique de la session.
4. Lancement automatique des services Grafana Prometheus.
5. Lancement automatiser les collecteurs de logs pour analyse calibrer dur les faiblesses connues de Jitsi-Meet.

## Jour-20 Tercium 5ième semaine | Réinstaller Prometheus | Création d'un dépôt GitHub pour par la suite Tests de charge / suivi des logs / maintenance du SI

### 1. Crédit à l'origine : [Guide & création](#)

```
jitsi-meet-infra/
├── .env.template
├── .env          # (non versionné)
├── .gitignore
├── Makefile
├── Makefile.test
├── README.md
├── SECURITY.md
├── CONFIGURATION.md
└── docs/
    └── Dépôt conseils et fichiers.docx
└── scripts/
    ├── automate_keys.sh
    ├── post-deploy.sh
    ├── push_to_infomaniak.sh
    └── rollback.sh
└── .github/
    └── workflows/
        ├── deploy.yml
        └── deploy-alt.yml  # (optionnel)
└── etc/
    ├── jitsi/
    ├── prosody/
    └── nginx/
└── opt/
    └── jitsi/
```

Voir Annexe Guide Github.

## 2. Finir l'installation de Prometheus.

La liste complète des fichiers Prometheus à restaurer ou réinstaller proprement, avec leurs emplacements standard sur un système Linux (ex : Ubuntu Server), structurée par rôle :

### FICHIERS ET DOSSIERS PROMETHEUS – STRUCTURE STANDARD

#### 1. Dossier principal de configuration

**Emplacement :**

/etc/prometheus/

Contenu essentiel :

Fichier	Rôle
<b>prometheus.yml</b>	Fichier principal de configuration (scrape configs, alert rules)
<b>alert.rules.yml</b>	(Facultatif) Règles d'alerte Prometheus si rule_files est utilisé
<b>blackbox.yml</b>	(Facultatif) Configuration de modules pour blackbox_exporter
<b>web.yml</b>	(Facultatif) Configuration du reverse proxy ou HTTPS/TLS

#### 2. Répertoires système à vérifier / créer

Répertoire	Rôle
/etc/prometheus/	Configs (décris ci-dessus)
/var/lib/prometheus/	Données (base TSDB locale)
/var/log/prometheus/	Logs (optionnel, si configuré manuellement)

#### 3. Exporters (agents de collecte de métriques)

**node\_exporter (système de base)**

- Fichier binaire : /usr/local/bin/node\_exporter
- Service systemd : /etc/systemd/system/node\_exporter.service

### **blackbox\_exporter (HTTP, ping, TCP...)**

- Binaire : /usr/local/bin/blackbox\_exporter
- Config : /etc/prometheus/blackbox.yml
- Service : /etc/systemd/system/blackbox\_exporter.service

### **telegraf (si utilisé pour enrichir les métriques)**

- Config : /etc/telegraf/telegraf.conf
- Service : telegraf.service

## **4. Dashboards Grafana (JSON)**

### **Si tu sauvegardes des dashboards :**

- Emplacement recommandé de versionnement :  
`/opt/prometheus-grafana-dashboards/`
- Exemple :
  - `dashboard_node_exporter.json`
  - `dashboard_blackbox_tcp_http.json`

### **Fichiers de service systemd à restaurer**

#### **Fichier systemd**

`/etc/systemd/system/prometheus.service`  
`/etc/systemd/system/node_exporter.service`  
`/etc/systemd/system/blackbox_exporter.service`

#### **Service contrôlé**

`Lancement Prometheus`  
`Lancement node_exporter`  
`Lancement blackbox_exporter`

### **Étapes de restauration minimale**

#### **# 1. Recréer le dossier**

```
sudo mkdir -p /etc/prometheus /var/lib/prometheus
```

#### **# 2. Copier les fichiers**

```
sudo cp prometheus.yml /etc/prometheus/  
sudo cp alert.rules.yml /etc/prometheus/  
sudo cp blackbox.yml /etc/prometheus/
```

#### **# 3. Vérifier les permissions**

```
sudo chown -R prometheus:prometheus /etc/prometheus /var/lib/prometheus
```

#### **# 4. Activer le service**

```
sudo systemctl daemon-reexec  
sudo systemctl restart prometheus.service  
sudo systemctl status prometheus
```

- 3. Puis finaliser le démarrage automatique de la session.**
- 4. Lancement automatique des services Grafana Prometheus.**
- 5. Lancement automatiser les collecteurs de logs pour analyser, calibrer détecter les faiblesses connues de Jitsi-Meet.**

## Jour-20 Tercium 5ième semaine | Réinstaller Prometheus | Création d'un multi dépôt GitHub | Installation de la suite de sécurité Tests de charge / suivi des logs / maintenance du SI

J'ai des snapshots sur l'instance cela serait judicieux de les rappeler pour relancer prometheus complétement sans tout réinstaller. Stupide de n'y pas avoir pensé plus tôt. En l'écrivant je réponds à la question, je suis bien un Igor du fameux Baron Franckenstein. Secondo je pourrais donc rapatrier ce snapshot dans le dépôt statique ou dynamique de github.

Raisonnement logique, l'on doit raisonner ici comme un architecte d'infrastructure pragmatique. Si un snapshot fonctionnel existe sur l'instance, alors il représente un point de restauration fiable, souvent plus rapide et cohérent qu'une réinstallation manuelle complète.

### Réponses directes à ces deux questions

#### 1. Relancer Prometheus depuis un snapshot ?

Oui, c'est pertinent.

#### Avantages :

- Gain de temps majeur (services, permissions, ports déjà validés)
- Réduction du risque d'erreur humaine
- Réactivation possible même sans réseau si snapshot local
- Compatible avec une relance post-nettoyage de dépendances conflictuelles (**Telegraf, Wazuh...**)

#### Procédure proposée :

- Restaurer uniquement les dossiers ciblés :
  - /etc/prometheus/
  - /opt/prometheus/ (*ou autre chemin si déplacé*)
- Redémarrer les services :  
**cd** : sudo systemctl daemon-reexec  
**cd** : sudo systemctl restart prometheus  
**cd** : sudo systemctl restart prometheus-node-exporter
- Vérifier :  
**cd** : curl -s http://localhost:9091/metrics | grep '^up'

#### S'assurer que :

- Les fichiers **prometheus.yml**, **alert.rules.yml** et **blackbox.yml** sont présents.
- L'utilisateur **prometheus** a bien les droits.
- Le port (**9091 ou autre**) est libre.

## 2. Exporter ce snapshot vers le dépôt GitHub ?

Oui, pour le dépôt statique uniquement.

Mais :

- **Jamais dans le dépôt dynamique, car :**
  - Il doit rester léger, lisible, CI/CD friendly
  - Des fichiers lourds / binaires casseraient la logique GitOps

Procédure :

- Monter le snapshot sur **/mnt/snapshot\_restore/**
- Copier uniquement les fichiers configurables :

```
cp /mnt/snapshot_restore/etc/prometheus/*.yml ./jitsi-meet-snapshot/prometheus_config/
cp /mnt/snapshot_restore/opt/prometheus/consoles/* ./jitsi-meet-snapshot/consoles/
```

- Ajouter un fichier **README\_RESTORE.md** dans **jitsi-meet-snapshot/** :

Ce snapshot a été extrait depuis l'instance Infomaniak via une restauration système (date, UUID, etc.). Les fichiers sont utilisables pour une réactivation complète de Prometheus sans réinstallation.

### Résumé opérationnel

Action	Dépôt concerné	But
Restaurer Prometheus via snapshot	Aucun (sur instance)	Réactiver la supervision rapidement
Exporter la conf vers GitHub	jitsi-meet-snapshot	Archiver proprement la version stable
Exclure ces fichiers du dynamique	jitsi-meet-infra	Ne pas surcharger le CI/CD pipeline