



Octubre de 2022

**Laboratorio 5**

Monitoreo y supervisión de redes

Datos Personales:

Nombre:

Número estudiante:

Fecha:

## INTRODUCCIÓN

### Topología

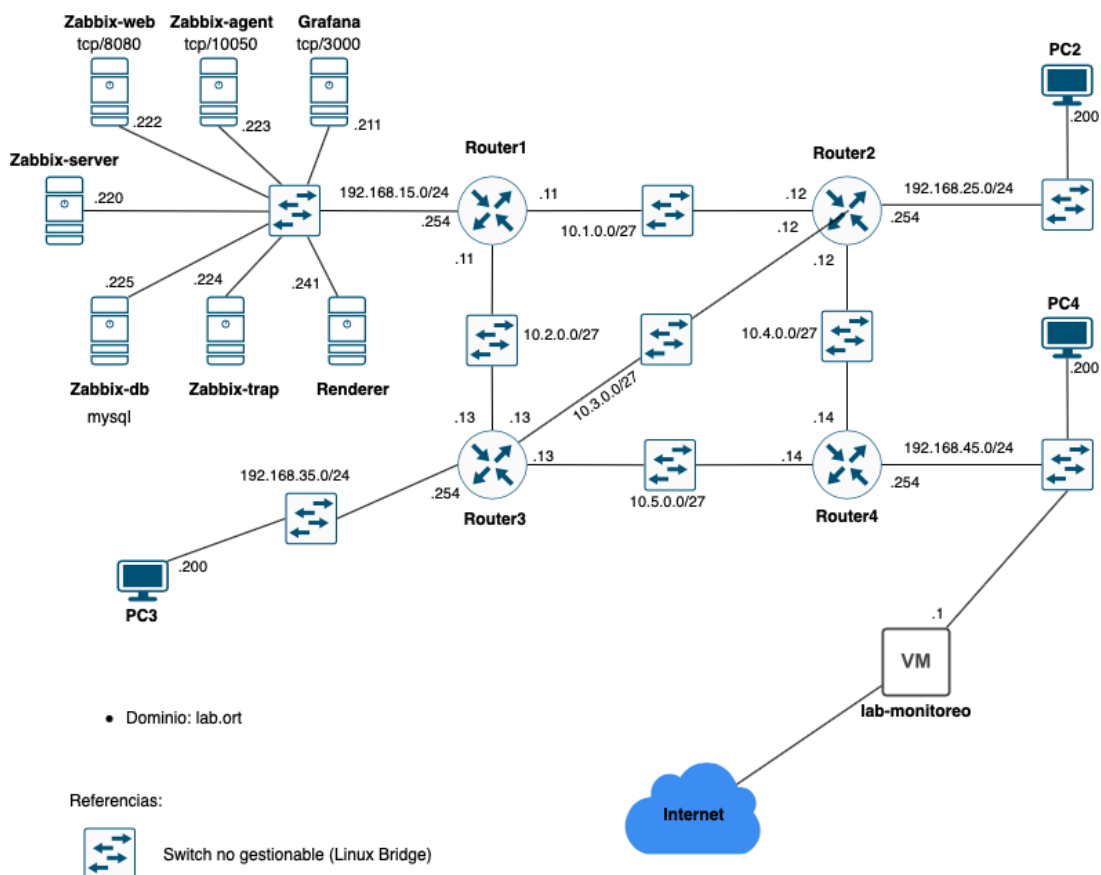


Imagen 1: Topología de red

La conexión se realiza mediante SSH a la máquina virtual lab-monitoreo. Por defecto se presenta en el puerto tcp/2222 de la interface de red de la computadora personal del estudiante

Usuario	estudiante
Password	estudiante
Puerto	2222

# PRÁCTICOS

## Práctico 1

### Reconocimiento de routers

Veremos como se conforma la red, en particular a nivel de ruteo

1. Conectarse por SSH a la VM lab-monitoreo con redirección X11 habilitada
2. Si ingresa con MobaXterm viene habilitado por defecto

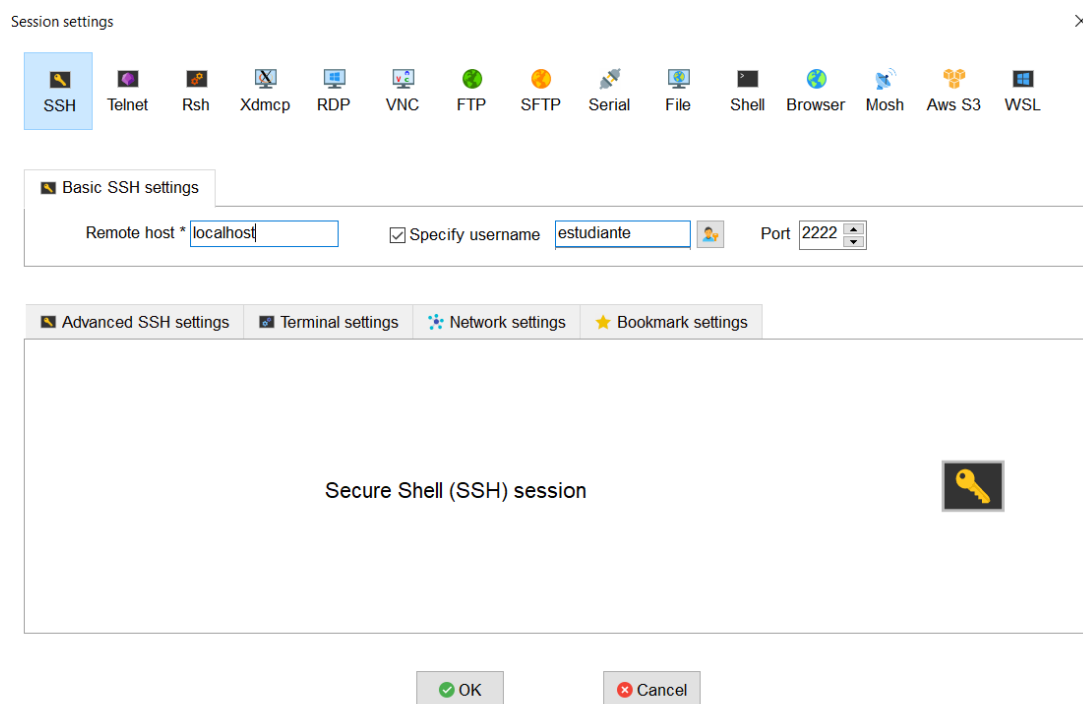


Imagen 2: Ejemplo en MobaXterm

3. Para habilitar X11 en Putty:

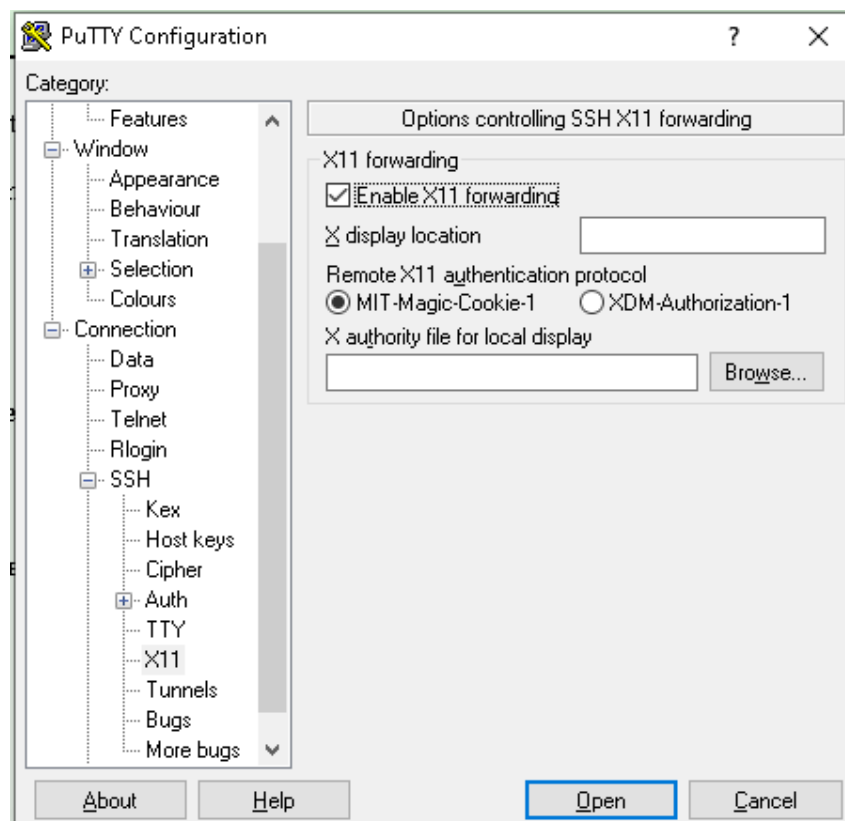


Imagen 3: Configuración Putty

4. Primero que nada prepararemos el ambiente para este laboratorio, ejecutemos:
  - *lab5*
5. En el siguiente diagrama se puede ver como se encuentran configurados los routers

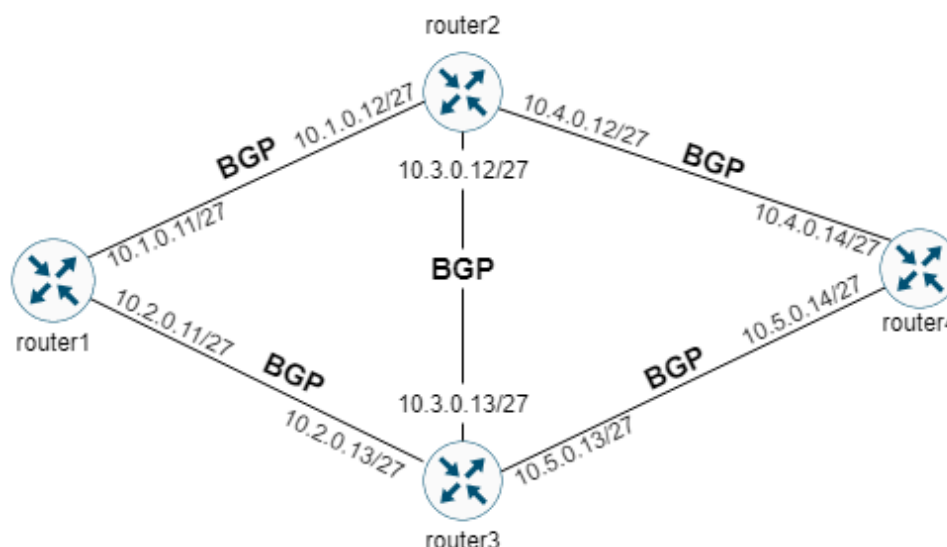


Imagen 4: Diagrama routers

6. Los hosts que actúan de routers están corriendo BIRD, un proyecto open source que permite configurar diferentes protocolos de ruteo. Por más detalle pueden ver la web oficial: <https://bird.network.cz/>
7. BIRD nos deja disponible una consola para interactuar con el plano de control
8. Ingresemos a router3.lab.ort
  - `ssh router3.lab.ort`
9. Ejecutemos `birdc` en el dispositivo para acceder a la consola
  - `birdc`
10. En este momento podemos utilizar algunos comandos para ver la configuración, con sintaxis similar a los de un router
  - `show interfaces`

```
bird> show interfaces
lo up (index=1)
    MultiAccess AdminUp LinkUp Loopback Ignored MTU=65536
    127.0.0.1/8 (Preferred, scope host)
eth2 up (index=17)
    MultiAccess Broadcast Multicast AdminUp LinkUp MTU=1500
    10.3.0.13/27 (Preferred, scope site)
eth3 up (index=31)
    MultiAccess Broadcast Multicast AdminUp LinkUp MTU=1500
    10.5.0.13/27 (Preferred, scope site)
eth0 up (index=41)
    MultiAccess Broadcast Multicast AdminUp LinkUp MTU=1500
    192.168.35.254/24 (Preferred, scope site)
eth1 up (index=49)
    MultiAccess Broadcast Multicast AdminUp LinkUp MTU=1500
    10.2.0.13/27 (Preferred, scope site)
```

Imagen 5: BIRD - show interfaces

- Con este comando podemos ver datos de las interfaces, como ser:
  - Nombre
  - Número de índice SNMP
  - Estado del enlace (up/down)
  - Estado administrativo (up/down)
  - MTU (Maximum Transfer Unit)
  - IP/Máscara
- `show protocols`

```
bird> show protocols
Name      Proto    Table      State    Since          Info
device1   Device   ---        up       01:24:08.984
direct1   Direct   ---        down     03:27:40.538
kernel1   Kernel   master4    up       01:24:08.984
STATIC4    Static   master4    up       01:24:08.984
R20515     BGP      ---        up       03:27:44.861   Established
R20525     BGP      ---        up       03:30:58.321   Established
R20545     BGP      ---        up       01:24:13.314   Established
```

Imagen 6: BIRD - show protocols

- Con este comando podemos ver datos de los protocolos de ruteo configurados, en particular nos interesa los de tipo BGP:
  - Nombre de la sesión BGP, definido en la configuración
  - State: up/start
  - Info: Connect/Idle/Active/Established (para que una sesión BGP funcione correctamente debe estar en estado Established)
- *show route*

```
bird> show route
Table master4:
0.0.0.0/0          unicast [R20545 01:24:13.314] * (100) [AS20545i]
  via 10.5.0.14 on eth3
  unicast [R20515 03:31:01.227] (100) [AS20545i]
  via 10.2.0.11 on eth1
  unicast [R20525 03:31:01.227] (100) [AS20545i]
  via 10.3.0.12 on eth2
10.3.0.0/27        unicast [STATIC4 01:24:08.984] * (110)
  dev eth2
  unicast [R20545 03:31:01.228] (100) [AS20525i]
  via 10.5.0.14 on eth3
  unicast [R20525 03:30:58.322] (100) [AS20525i]
  via 10.3.0.12 on eth2
  unicast [R20515 03:27:44.861] (100) [AS20525i]
  via 10.2.0.11 on eth1
192.168.26.0/24    unicast [R20525 03:30:58.322] * (100) [AS20525i]
  via 10.3.0.12 on eth2
  unicast [R20545 03:31:01.228] (100) [AS20525i]
  via 10.5.0.14 on eth3
  unicast [R20515 03:27:44.861] (100) [AS20525i]
  via 10.2.0.11 on eth1
10.1.0.0/27        unicast [R20515 03:27:44.861] * (100) [AS20515i]
  via 10.2.0.11 on eth1
  unicast [R20545 03:31:01.228] (100) [AS20525i]
  via 10.5.0.14 on eth3
  unicast [R20525 03:30:58.322] (100) [AS20525i]
  via 10.3.0.12 on eth2
```

Imagen 7: BIRD - show routes

- Con este comando podemos ver la tabla de rutas:

- Prefijo
- Próximos saltos disponibles (el marcado con \* es el que está activo)
- Tiempo que hace que está el prefijo en la tabla de rutas

11. En el navegador de nuestra PC

- Ingrese a <http://localhost:2223>
- Credenciales de acceso:

Usuario	Admin
Password	zabbix

12. Verifiquemos que los hosts router1, router2, router3 y router4 tengan asociado el template "Template OS Linux SNMP"

13. Verifiquemos que no contamos con problemas activos en Zabbix

14. Ejecutemos lab5.1

- *lab5.1*
- Este lab provocará:
  - la caída de 1 interfaz que conecta Router3 con Router2
  - la caída de 1 interfaz que conecta Router4 con Router2

15. Esperemos un par de minutos y revisemos los eventos en Zabbix

16. NOTA: Puede notar una indisponibilidad de Zabbix durante la convergencia de ruteo  
**¿Observa caída de interfaces?**

17. Tome una captura de los problemas reportados por Zabbix para registro

18. Con los comandos vistos anteriormente repase como se observa la conectividad entre los routers

**¿qué cambios percibe? ¿en qué routers?**

19. Para restablecer la conectividad ejecute lab5.1.restore
  - *lab5.1.restore*
20. Verifiquemos que los problemas en Zabbix se restablecieron

## Práctico 2

### Importacion de configuracion

Este laboratorio tiene como finalidad ver cómo se puede importar configuración a partir de la cual se realice un monitoreo a medida de las sesiones BGP existentes en la red

1. Nos basamos en el siguiente proyecto existente en Github, con algunas modificaciones menores: <https://github.com/verovd/workzabbix/tree/master/zabbix-bird-bgp-zabbix-bird-bgp>
2. La configuración consta de dos partes:
  - Copiar scripts y archivos de configuración que deben estar disponibles en un directorio previsto por Zabbix Agent para que implemente el monitoreo
  - Importar la configuración de items, triggers y demás componentes en la web de Zabbix
3. El primer punto se ha dado resuelto, ya se encuentran copiados los archivos en los directorios correspondientes de cada router
4. Para entender mejor los cambios que se harán observemos los archivos que se encuentran en /home/estudiante/lab-monitoreo/LAB5/
5. Para importar un nuevo template vayamos a Configuration->Templates->Import
6. Busquemos el archivo "bird monitoring zabbix.xml" en el directorio mencionado anteriormente

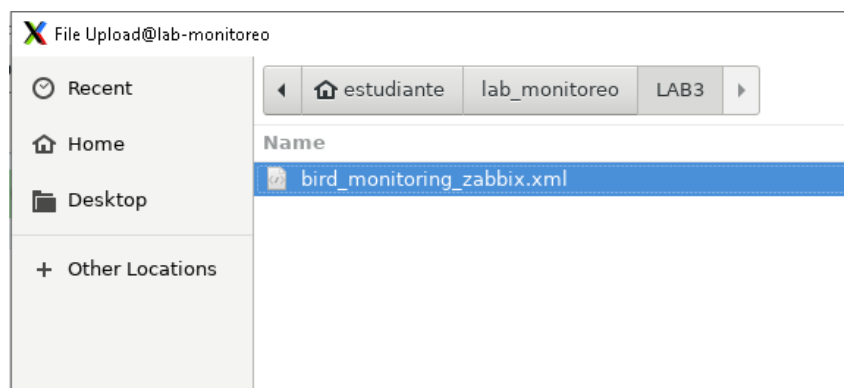


Imagen 8: Zabbix Import Configuration

7. Marquemos todas las casillas de la columna "Create new" y marquemos Import



\* Import file

Rules	Update existing	Create new	Delete missing
Groups		<input checked="" type="checkbox"/>	
Hosts	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Templates	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Template screens	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Template linkage		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Applications		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Items	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Discovery rules	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Triggers	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Graphs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Web scenarios	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Screens	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Maps	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Images	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Media types	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Value mappings	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

Imagen 9: Zabbix Import Configuration

8. Vinculemos el nuevo template a los routers (template "bird bgp")
9. Luego que se encuentre todo configurado repitamos la prueba de laboratorio 5.1
10. Ejecutemos lab5.1
  - lab5.1
11. Esperemos un par de minutos y revisemos los eventos en Zabbix
12. Puede notar una indisponibilidad de Zabbix durante la convergencia de ruteo

**¿Observa más alarmas que la vez anterior?**

### Tome registro de las alarmas que ve activas

13. Volvamos el laboratorio a la normalidad, ejecute lab5.1.restore

- *lab5.1.restore*

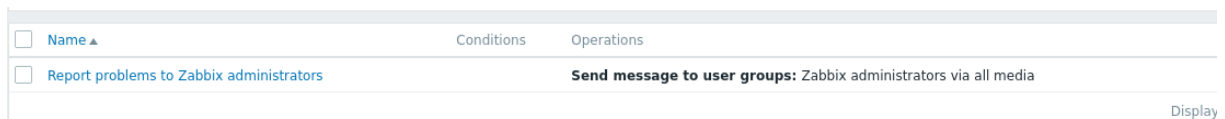
14. Si por algún motivo no funciona o desea probar otras configuraciones puede repetir las acciones lab5.1 y lab5.1.restore según quiera generar el problema en la red, o solucionarlo

## Práctico 3

### Envío de notificaciones

El objetivo de este laboratorio es configurar el envío de notificaciones ante problemas, veremos un ejemplo con Telegram

1. Vayamos a Configuration->Actions para habilitar el envío de notificaciones

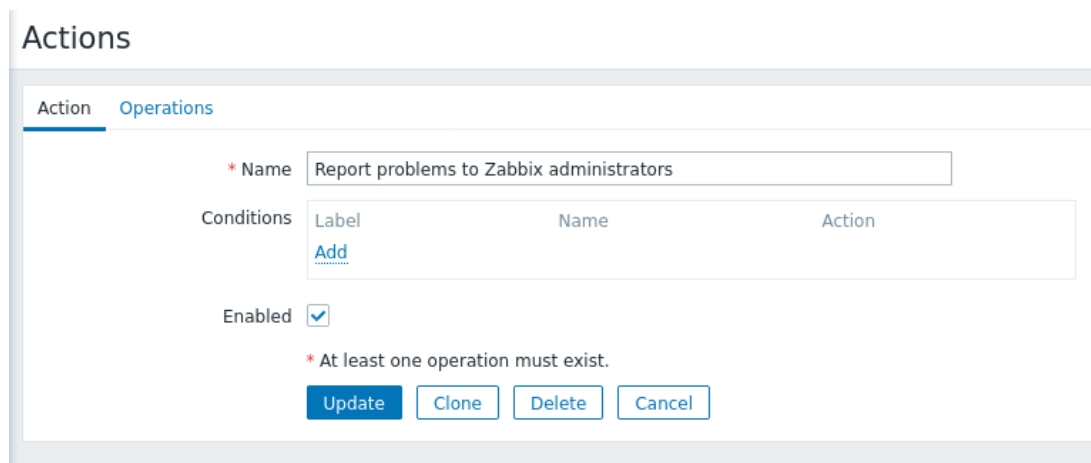


<input type="checkbox"/> Name ▲	Conditions	Operations
<input type="checkbox"/> Report problems to Zabbix administrators		Send message to user groups: Zabbix administrators via all media

Display

Imagen 10: Zabbix Configuration Actions

2. Editemos Report problems to Zabbix administrators



### Actions

**Action** **Operations**

\* Name

Conditions

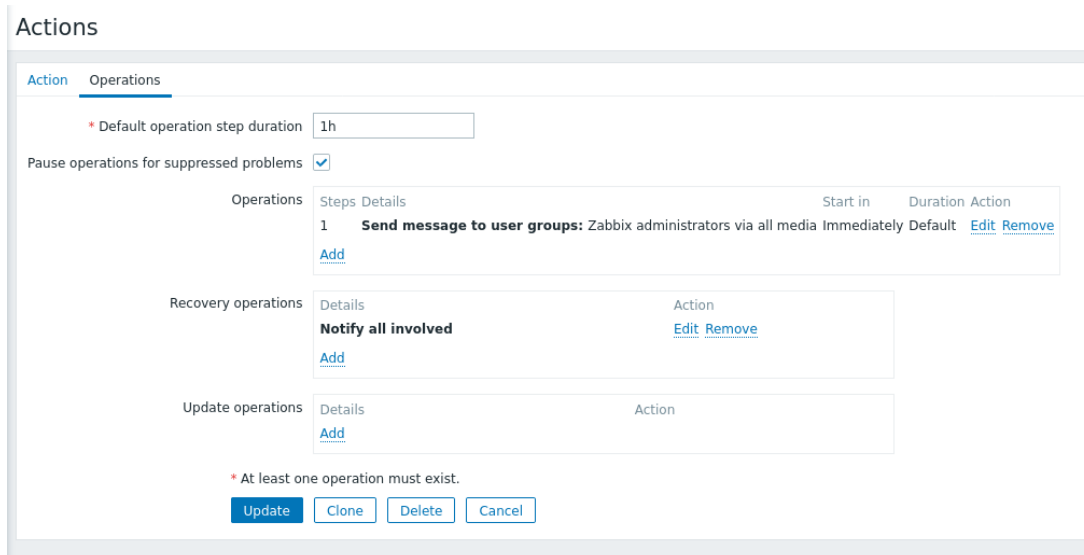
Label	Name	Action
<a href="#">Add</a>		

Enabled ☒

\* At least one operation must exist.

Imagen 11: Zabbix Configuration Actions

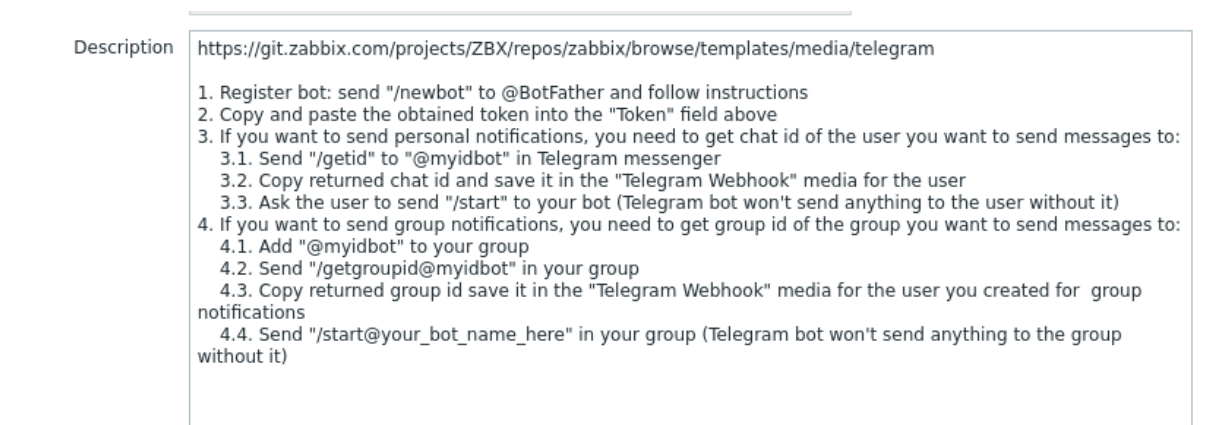
- Si accedemos a la pestaña operations podremos ver el tipo de evento por el que notificará y a qué usuarios



The screenshot shows the 'Actions' configuration page in Zabbix. The 'Operations' tab is active. At the top, there is a field for 'Default operation step duration' set to '1h' and a checkbox for 'Pause operations for suppressed problems' which is checked. Below this, there are three sections: 'Operations', 'Recovery operations', and 'Update operations'. The 'Operations' section contains a table with one row: '1 Send message to user groups: Zabbix administrators via all media' with 'Start in: Immediately', 'Duration: Default', and 'Action: Edit Remove'. The 'Recovery operations' section contains a table with one row: 'Notify all involved' with 'Action: Edit Remove'. The 'Update operations' section is empty. At the bottom, there is a note: '\* At least one operation must exist.' and buttons for 'Update', 'Clone', 'Delete', and 'Cancel'.

Imagen 12: Zabbix Configuration Actions

- A continuación debemos configurar un canal de notificación, ingresando a Administration->Media Types
- Ingrese a editar el media type definido para Telegram
- Se puede ver que en el campo Description hay un procedimiento de como configurarlo



The screenshot shows the 'Description' field for a Telegram media type in Zabbix. The field contains a URL and a list of steps for configuring the Telegram bot and webhooks.

https://git.zabbix.com/projects/ZBX/repos/zabbix/browse/templates/media/telegram

1. Register bot: send "/newbot" to @BotFather and follow instructions
2. Copy and paste the obtained token into the "Token" field above
3. If you want to send personal notifications, you need to get chat id of the user you want to send messages to:
  - 3.1. Send "/getid" to "@myidbot" in Telegram messenger
  - 3.2. Copy returned chat id and save it in the "Telegram Webhook" media for the user
  - 3.3. Ask the user to send "/start" to your bot (Telegram bot won't send anything to the user without it)
4. If you want to send group notifications, you need to get group id of the group you want to send messages to:
  - 4.1. Add "@myidbot" to your group
  - 4.2. Send "/getgroupid@myidbot" in your group
  - 4.3. Copy returned group id save it in the "Telegram Webhook" media for the user you created for group notifications
  - 4.4. Send "/start@your\_bot\_name\_here" in your group (Telegram bot won't send anything to the group without it)

Imagen 13: Zabbix Telegram Description

- Podemos ingresar a Telegram desde la web <https://web.telegram.org>

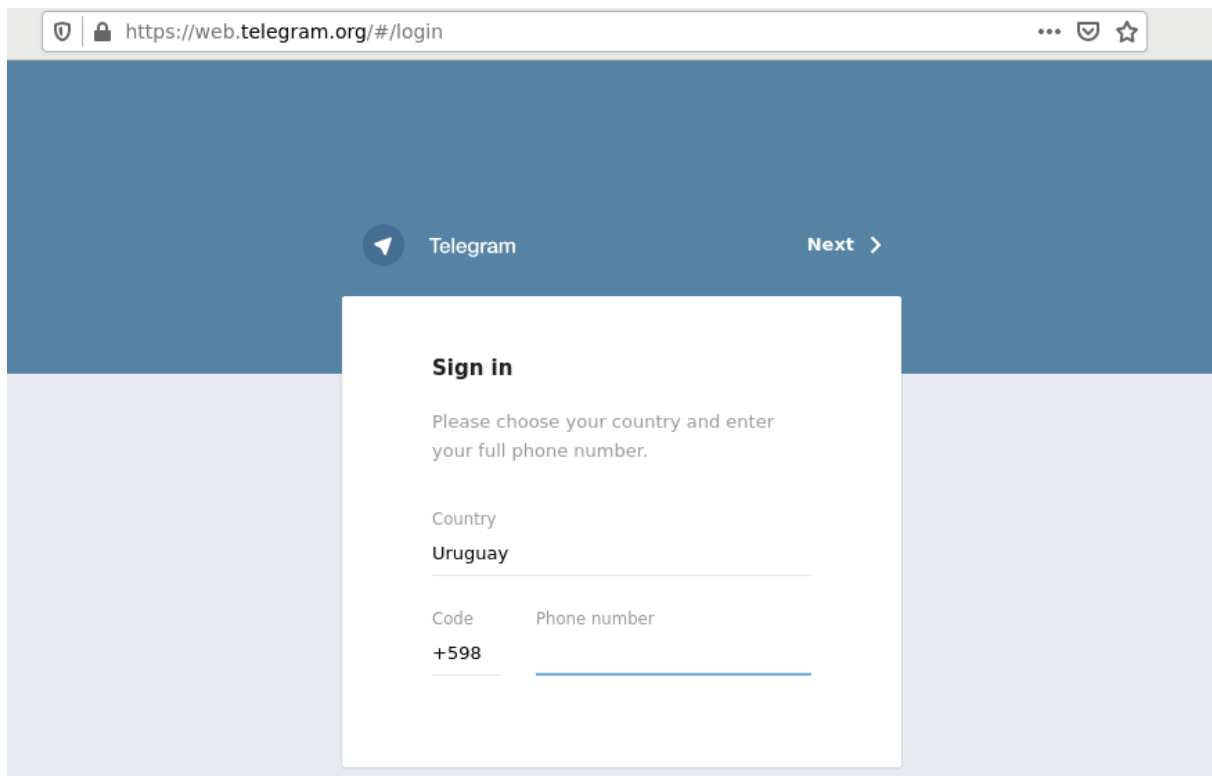


Imagen 14: Telegram Login

8. En caso de no contar con una cuenta se puede crear facilmente de forma gratuita
9. Luego de acceder, buscaremos el usuario BotFather

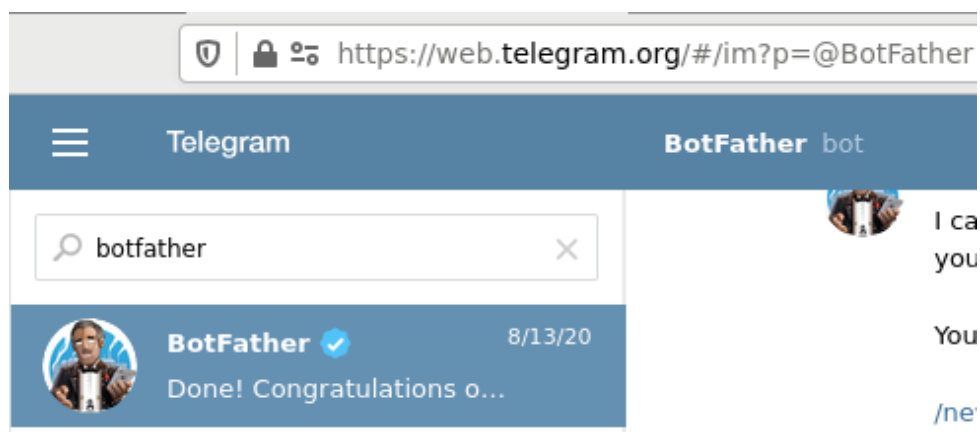


Imagen 15: Telegram Login

10. Para crear el bot que utilizaremos a la hora de notificar, debemos:
  - Iniciar el bot: /start
  - Solicitar la creación de un bot: /newbot
  - Asignarle un nombre

- Asignarle un nombre de usuario (debe terminar en "bot")

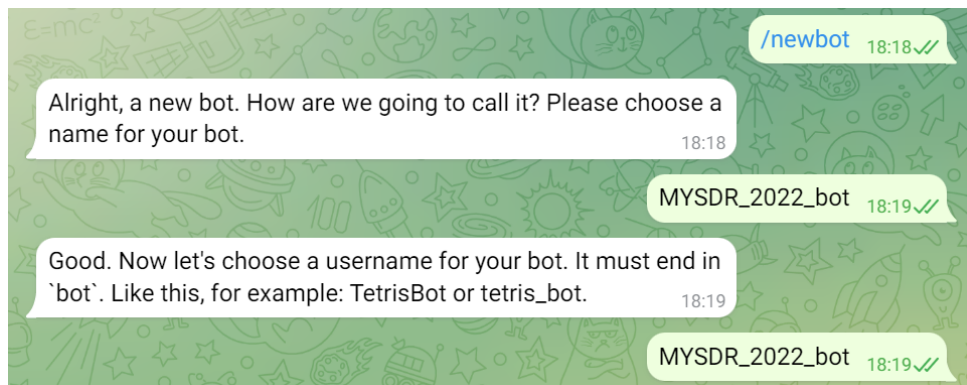


Imagen 16: Telegram BotFather

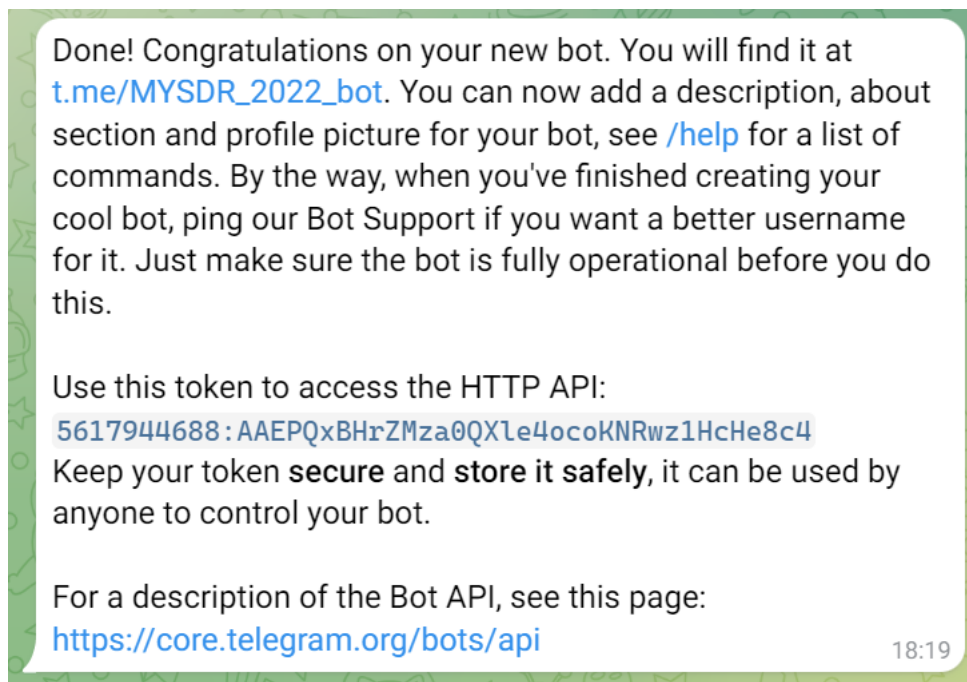


Imagen 17: Telegram BotFather

11. El token que fue generado para nuestro bot lo debemos copiar y pegar en el campo Token, dentro de los parámetros de nuestro Media Type de Telegram

## Media types

Media type   **Message templates**   Options

---

\* Name

Type

Parameters

Name	Value	Action
<input type="text" value="Message"/>	<input type="text" value="{ALERT.MESSAGE}"/>	<a href="#">Remove</a>
<input type="text" value="ParseMode"/>	<input type="text"/>	<a href="#">Remove</a>
<input type="text" value="Subject"/>	<input type="text" value="{ALERT.SUBJECT}"/>	<a href="#">Remove</a>
<input type="text" value="To"/>	<input type="text" value="{ALERT.SENDTO}"/>	<a href="#">Remove</a>
<input type="text" value="Token"/>	<input type="text" value="5617944688:AAEPQxBHrZMza0C"/>	<a href="#">Remove</a>
<a href="#">Add</a>		

\* Script

Timeout

Imagen 18: Zabbix Telegram Token

12. Ahora debemos obtener el ID de un usuario valido de Telegram al que querramos que se notifique, para eso busquemos el usuario myidbot

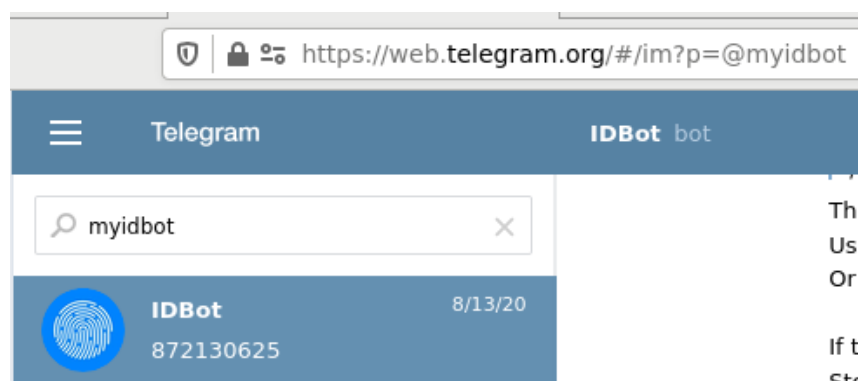


Imagen 19: Telegram myidbot

13. Iniciemos el bot

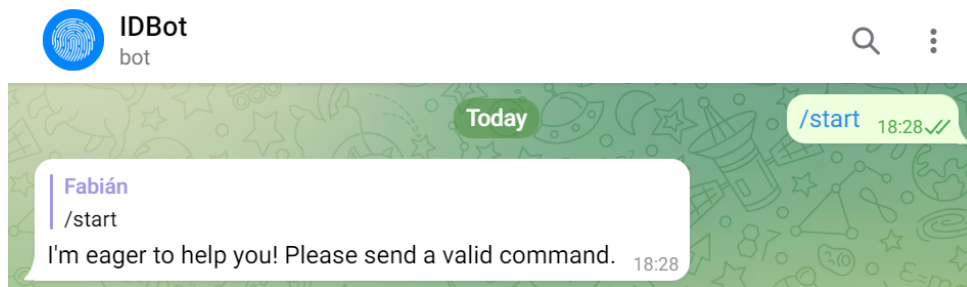


Imagen 20: Telegram myidbot

14. Solicitemos nos muestre nuestro ID

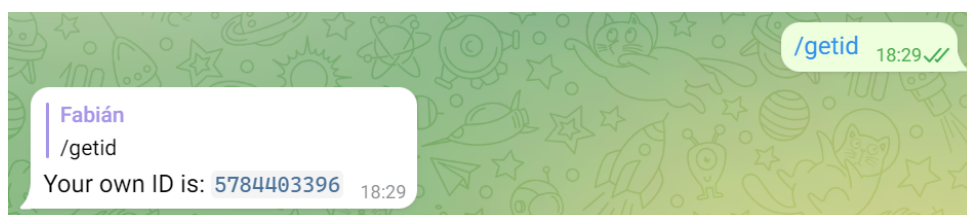


Imagen 21: Telegram myidbot

15. Vayamos a Administration->Users->Create user

- Crearemos un usuario con las siguientes características (para que lleguen las notificaciones al docente):
- Alias: docente\_MYSDR
- Group: Zabbix administrators

## Users

User Media Permissions

\* Alias

Name

Surname

\* Groups    
type here to search

Password

Language

Theme

Auto-login ☐

Auto-logout ☐

\* Refresh

\* Rows per page

URL (after login)

Imagen 22: Zabbix Create User

- Media->Add

Media

Type

\* Send to

\* When active

Use if severity ☒ Not classified  
☒ Information  
☒ Warning  
☒ Average  
☒ High  
☒ Disaster

Enabled ☒

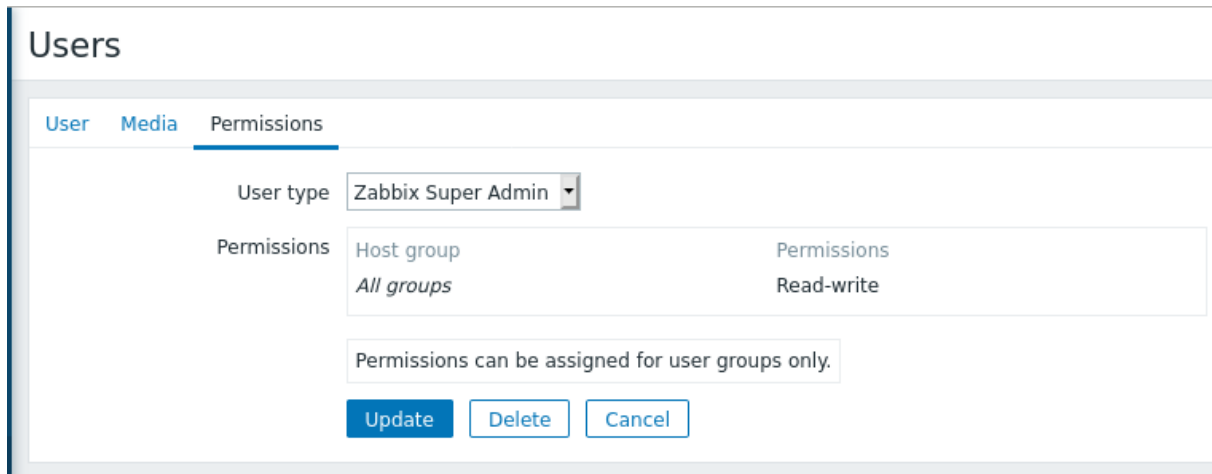
Imagen 23: Zabbix Create User

- Type: Telegram



- Send to: 5784403396
- When Active: 1-7,00:00-24:00
- Enabled: Si

#### 16. Permissions



**Users**

User Media **Permissions**

User type: Zabbix Super Admin

Host group	Permissions
All groups	Read-write

Permissions can be assigned for user groups only.

Update Delete Cancel

Imagen 24: Zabbix Create User

- User type: Zabbix Super Admin
- Permissions: All groups / Read-write

#### 17. Crear un nuevo usuario, con las mismas características excepto:

- Alias: Nombre de usuario personal
- Media->Send to: ID obtenido en el bot de Telegram myidbot

#### 18. A continuación deberemos:

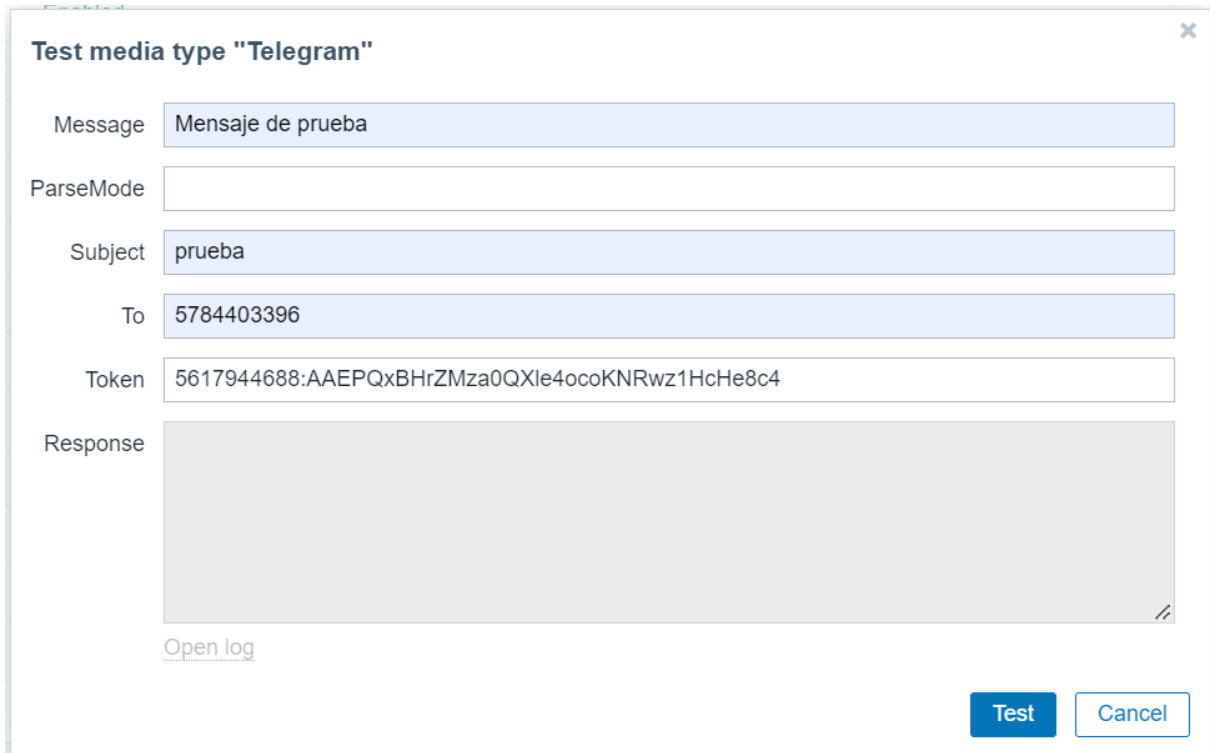
- Buscar en Telegram el bot que hemos creado dentro de los usuarios
- Iniciar el bot

#### 19. En este momento estamos en condiciones de que las notificaciones sean enviadas

## Práctico 4

### Probemos el envío de notificaciones

1. Vayamos a Media types->Telegram->Test



**Test media type "Telegram"**

Message: Mensaje de prueba

ParseMode:

Subject: prueba

To: 5784403396

Token: 5617944688:AAEPQxBHrZMza0QXle4ocoKNRwz1HcHe8c4

Response:

[Open log](#)

Test Cancel

Imagen 25: Zabbix Telegram test

- Message: Contenido del mensaje
  - Subject: Asunto
  - To: ID obtenido con myidbot
2. Verifiquemos que el mensaje llega a nuestra cuenta de Telegram
  3. Probemos con mensajes reales, ejecutemos lab5.2
    - *lab5.2*
    - Este lab provocará:
      - la caída de 1 interfaz que conecta Router3 con Router1
  4. Para recuperar el ambiente ejecutemos lab5.2.restore
    - *lab5.2.restore*