ZABBIX

Hands-On with zabbix_utils

https://github.com/knaglis/workshop-benelux

Kristaps Naglis

Integration Engineer

Environment preparation

Installing zabbix_utils (from package manager)

Additional dependency installation (from PyPI)

Getting code base from git

Working with zabbix_utils

Configure Zabbix instance for API access

Test if API works

Writing requests utilizing AsyncZabbixAPI

Testing Zabbix user access scope from API

Parsing the received data

How to handle CPU intensive code vs I/O intensive code

Z A B B I X

ZABBIX

Connect to your VM

Open terminal and log into your VM using SSH. For this, use the previously shared details about your VM

```
$ ssh root@student-XX-ws3.zabbix.training
The authenticity of host student-XX-ws3.zabbix.training (A.B.C.D)' can't be
established.
ED25519 key fingerprint is SHA256:h8lirvWk5Z/amgFXHSiPyNiH6Wwr6j/bT1B4WkSnaag.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'student-XX-ws3.zabbix.training' (ED25519) to the list
of known hosts.
root@student-XX-ws3.zabbix.training's password:
Web console: https://student-XX:9090/ or https://A.B.C.D:9090/

Last failed login: Wed Feb 19 17:02:21 UTC 2025 from 101.89.134.212 on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Wed Feb 19 17:01:21 2025 from A.B.C.D
[root@student-01 ~]#
```

ZABBIX

Getting the required tools

Setup Zabbix package repository

```
$ rpm -Uvh \
https://repo.zabbix.com/zabbix/7.0/centos/9/x86_64/zabbix-release-latest-7.0.el9.noarch.rpm

  Retrieving https://repo.zabbix.com/zabbix/7.0/centos/9/x86_64/zabbix-release-latest-
  7.0.el9.noarch.rpm
  Verifying...                          ################################# [100%]
  Preparing...                          ################################# [100%]
  Updating / installing...
     1:zabbix-release-7.0-5.el9         ################################# [100%]

$ dnf clean all
```

Install git and Zabbix_utils from package manager

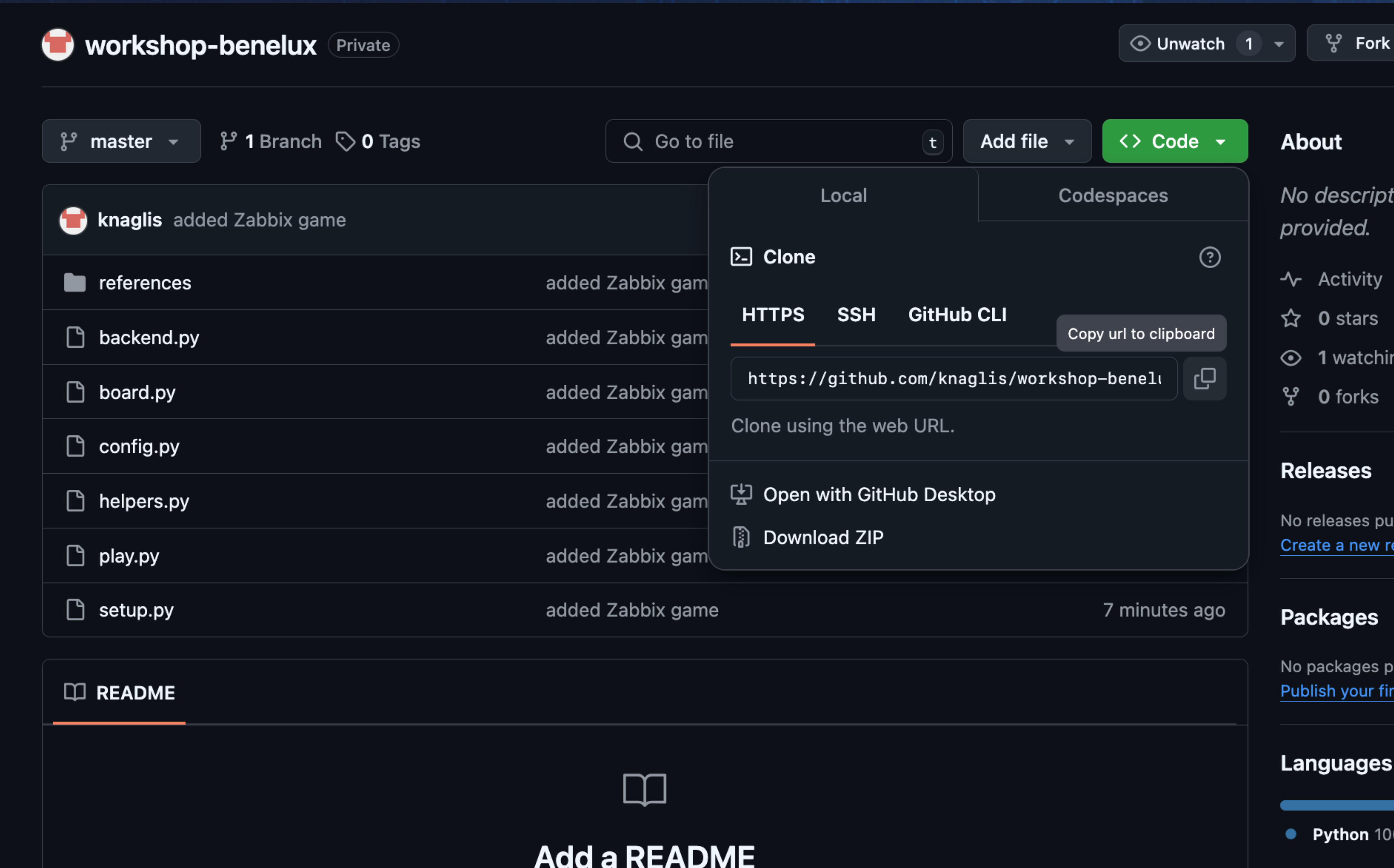
```
$ dnf install git python3-zabbix-utils

  Dependencies resolved.
  ===============================================================================
   Package                Architecture    Version        Repository          Size
  ===============================================================================
  Installing:
   git                    x86_64          2.47.1-1.el9   appstream           51 k
   python3-zabbix-utils   noarch          2.0.2-1        zabbix-tools        49 k
  Installing dependencies:
  ...
  Total download size: 8.0 M
  Installed size: 40 M
  Is this ok [y/N]: y
  Downloading Packages:
  ...
  Zabbix Official Repository (tools) - x86_64
  3.0 MB/s | 3.1 kB     00:00
  Importing GPG key 0x227618D8:
   Userid     : "Zabbix Tools <packager@zabbix.com>"
   Fingerprint: 8416 2C39 F000 750D DAEB 8163 05A7 FAEC 2276 18D8
   From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-TOOLS
  Is this ok [y/N]: y
  ...
```

Install aiohttp Python library from PyPI

```
$ pip3 install aiohttp
```

Clone the codebase from git

```
$ cd ~
$ git clone https://github.com/knaglis/workshop-benelux
$ cd workshop-benelux
```



workshop-benelux Private

Unwatch 1 | Fork

master | 1 Branch | 0 Tags | Go to file | Add file | Code

knaglis added Zabbix game

```
references     added Zabbix gam
backend.py     added Zabbix gam
board.py       added Zabbix gam
config.py      added Zabbix gam
helpers.py     added Zabbix gam
play.py        added Zabbix gam
setup.py       added Zabbix game          7 minutes ago
```

Local | Codespaces

Clone

HTTPS | SSH | GitHub CLI

Copy url to clipboard

```
https://github.com/knaglis/workshop-benelı
```

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

About

No descript provided.

Activity

0 stars

1 watchir

0 forks

Releases

No releases pu

Create a new r

Packages

No packages p

Publish your fir

Languages

Python 10

README

Add a README

ZABBIX

Project structure

backend.py	Manages Zabbix server-side updates of items
board.py	Support script to build game map/board
config.py	Holds configuration variables
helpers.py	Code that is required for proper functioning of game
play.py	Actual script to play the game
setup.py	Creates all required resources on Zabbix (users, groups, hosts, items, player tokens, connector, etc.)
playerTokens	Holds tokens of all players (generated only after executing setup.py)
references/	Directory that includes fully completed files we will be working on in this workshop. **If needed, use these files for reference and/or hints.**
play.py	Completed version of play.py

ZABBIX

Configure Zabbix for API access

- Log into your Zabbix instance

  http://student-XX-ws3.zabbix.training/zabbix

  username: Admin
  password: zabbix

- Change the password of Zabbix **Admin** account

  Navigate **Users -> Users -> Admin** and change the password

- Generate API access token for **Admin** account

  Navigate **Users -> API tokens** and **Create API token**

- Make sure to **save the generated** token and click **close**

ZABBIX

Code editing

In general, there are 3 options:

- Use any CLI text editor on the VM itself, such as Vim / Vi or Nano

- Use VSCode or forks of it with SSH remote connection to VM **(recommended option)**

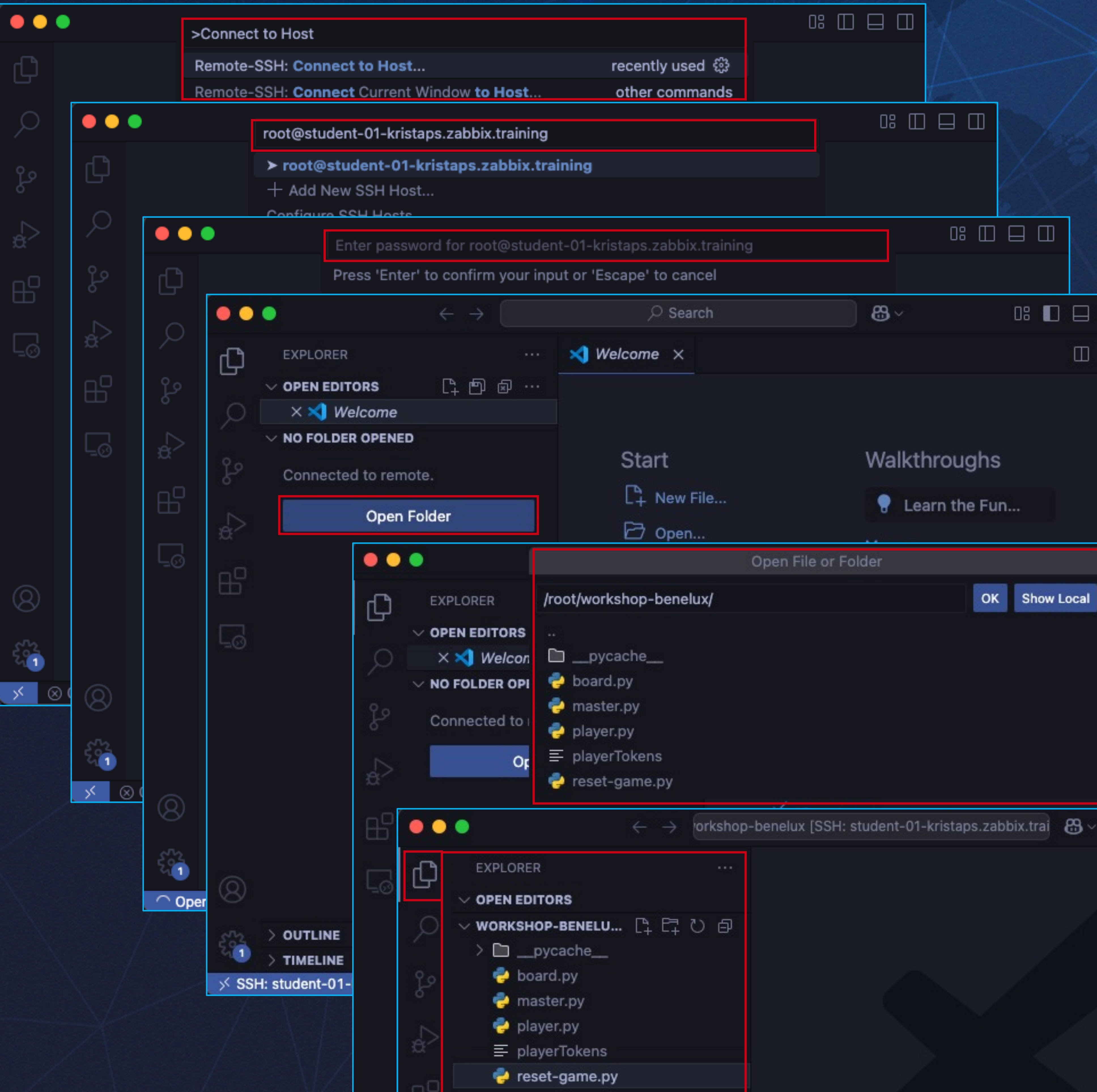
- Use any code editor on local PC and copy-paste code to VM.

  In this case, using Vim on VM will be useful, as Vim keyboard shortcut **ggdG** will delete all file contents and you can paste the edited file easily into VM.

All files use indentation of 2 SPACES

VSCode setup

CTRL + Shift + P (Windows/Linux) or **CMD + Shift + P** (Mac)



```
>Connect to Host
Remote-SSH: Connect to Host...                        recently used
Remote-SSH: Connect Current Window to Host...         other commands
```

```
root@student-01-kristaps.zabbix.training
> root@student-01-kristaps.zabbix.training
+ Add New SSH Host...
Configure SSH Hosts...
```

```
Enter password for root@student-01-kristaps.zabbix.training
Press 'Enter' to confirm your input or 'Escape' to cancel
```

```
EXPLORER
OPEN EDITORS
  × Welcome
NO FOLDER OPENED
  Connected to remote.
  [Open Folder]
```

```
Open File or Folder
/root/workshop-benelux/          [OK] [Show Local]
..
__pycache__
board.py
master.py
player.py
playerTokens
reset-game.py
```

```
orkshop-benelux [SSH: student-01-kristaps.zabbix.trai
EXPLORER
OPEN EDITORS
WORKSHOP-BENELU...
  > __pycache__
    board.py
    master.py
    player.py
    playerTokens
    reset-game.py
```

ZABBIX

config.py

Updating admin token

The previously generated token for admin account needs to be entered in the **zabbixAdminToken** variable. By default, it will be an empty string.

```
zabbixServerIP = '127.0.0.1'
zabbixServerPort = 10051
zabbixAPIPath = f'{zabbixServerIP}/zabbix'
zabbixAdminToken = ''
zabbixPlayerToken = ''
```

```
zabbixServerIP = '127.0.0.1'
zabbixServerPort = 10051
zabbixAPIPath = f'{zabbixServerIP}/zabbix'
zabbixAdminToken = 'your_admin_token'
zabbixPlayerToken = ''
```

ZABBIX

setup.py

Get the player token

Execute setup.py

```
$ ./setup.py -f
Getting users
Removing users
Player users removed

Getting players role
Removing players role
Players role removed

Adding players role
Players role added

Adding users
        Adding player users
Player users added

...

Adding connectors
Done adding connectors
```

After successful execution, there should appear a file named “**playerTokens**”. This file has all the access tokens for players that were generated during script execution.

```
$ cat playerTokens
{"Player 1":
"f74f4aa00f7623ef4c28bdd6692d0b4f81bc23d86ca294dd7b0cebc6eb3c1075"}
```

This token will be needs to be added in **config.py** variable **zabbixPlayerToken**. If multiple players were defined, send the respective token to each player.

ZABBIX

config.py

Updating player token

The previously generated token for player account needs to be entered in the zabbixPlayerToken variable. By default, it will be an empty string.

```
zabbixServerIP = '127.0.0.1'
zabbixServerPort = 10051
zabbixAPIPath = f'{zabbixServerIP}/zabbix'
zabbixAdminToken = 'your_admin_token'
zabbixPlayerToken = ''
```

```
zabbixServerIP = '127.0.0.1'
zabbixServerPort = 10051
zabbixAPIPath = f'{zabbixServerIP}/zabbix'
zabbixAdminToken = 'your_admin_token'
zabbixPlayerToken = 'your_player_token'
```

Enable connectors

This script uses Zabbix streaming protocol to receive map data from Zabbix, therefore connectors need to be enabled on server.

```
$ vim /etc/zabbix/zabbix_server.conf
```

```
...
### Option: StartConnectors
#       Number of pre-forked instances of connector workers.
#               The connector manager process is automatically started when
connector worker is started.
#
# Mandatory: no
# Range: 0-1000
# Default:
StartConnectors=3
...
```

```
$ systemctl restart zabbix-server
```

ZABBIX

play.py

AsyncIO library

The play.py file uses asyncio python library to make asynchronous requests to Zabbix, therefore at the top of the file, **AsyncZabbixAPI** from **zabbix_utils** needs to be imported

```
from zabbix_utils import AsyncZabbixAPI
```

play.py

Missing code

In general, play.py has 2 main components – UI and Zabbix API. Both components are split into separate classes – **UI()** and **Zabbix()**, respectively. Both are managed by **Game()** class. There is missing code in both **Zabbix(), Game()** classes and the **main** function, marked with comments starting with - *! IMPLEMENT*

```python
class Zabbix(AsyncMixin):
  async def __ainit__(self):
   # ! IMPLEMENT – asynchronous Zabbix API object
   # ! IMPLEMENT – asynchronous Zabbix sender object
  ...
  # ! IMPLEMENT – API login method
  ...
  # ! IMPLEMENT – request current player position
  ...
  # ! IMPLEMENT – request player host
  ...
  # ! IMPLEMENT – request score
  ...
  async def move(self, direction):
    ...
    # ! IMPLEMENT – asynchronous Zabbix sender to send updated position values
...
class Game(AsyncMixin):
  ...
  async def run(self):
    # ! IMPLEMENT – launching multiple functions with asyncio
    ...
   ...

async def runGame():
  game = await Game()
  await game.run()

if __name__ == "__main__":
  try:
    """"""
    # ! IMPLEMENT – run the asynchronous coroutine
```

ZABBIX

Where to start

https://www.zabbix.com/documentation/current/manual/api/reference

ZABBIX

play.py

Running asyncio coroutine

Asyncio coroutine should be started with **asyncio.run()**

```
async def runGame():
  game = await Game()
  await game.run()

if __name__ == "__main__":
  try:
    """"""
    # ! IMPLEMENT - run the asynchronous coroutine
    asyncio.run(runGame())
  except Exception as e:
    print(f'[ EXCEPTION ]: {e}')
    exit(1)
```

play.py

Fixing Zabbix() class

Create asynchronous Zabbix API object

```python
class Zabbix(AsyncMixin):
  ...
  async def __ainit__(self):
    # Initialize asynchronous defaults
    # ! IMPLEMENT - asynchronous Zabbix API object
    self.api = AsyncZabbixAPI(url=config.zabbixAPIPath, validate_certs=False)
    await self.apiLogin()
```

Create api login function

```python
class Zabbix(AsyncMixin):
  ...
  # ! IMPLEMENT - API login method
  async def apiLogin(self):
    await self.api.login(token=config.zabbixPlayerToken)
```

Let’s test if API works with player token

```python
class Zabbix(AsyncMixin):
  ...
  # ! IMPLEMENT - API login method
  async def apiLogin(self):
    await self.api.login(token=config.zabbixPlayerToken)
    print(f'Zabbix hosts: {await self.api.host.get(output=['host'])}')
    await self.api.logout()
    exit(0)
```

```
$ ./play.py
Zabbix hosts: [{'hostid': '10990', 'host': 'Player 1'}, {'hostid': '10991',
'host': 'Main Game'}]
```

play.py

Fixing Zabbix() class

Remove the last 3 added lines of code from the last step (print(), logout() and exit()), so the **apiLogin()** function looks like this:

```python
class Zabbix(AsyncMixin):
  ...
  # ! IMPLEMENT - API login method
  async def apiLogin(self):
    await self.api.login(token=config.zabbixPlayerToken)
```

However, we still need to implement some functions for the code to run. This function will get ID of the player host

```python
class Zabbix(AsyncMixin):
  def __init__(self):
    ...
    self.playerHostId = None
    self.playerHostName = None
  ...
  # ! IMPLEMENT - request player host
  async def setPlayerHost(self):
    hosts = await self.api.host.get(
      search={'host': 'Player'},
      output=['host', 'hostid']
    )
    self.playerHostId = hosts[0]['hostid']
    self.playerHostName = hosts[0]['host']

```

play.py

Fixing Zabbix() class

Test the API output

```
# ! IMPLEMENT - request player host
  async def setPlayerHost(self):
    hosts = await self.api.host.get(
      output=['host', 'hostid']
    )
    print(hosts)
    self.api.logout()
    exit(0)
    for host in hosts:
      if str(host['host']).startswith('Player'):
        self.playerHostId = host['hostid']
        self.playerHostName = host['host']
```

```
$ ./play.py
[{'hostid': '10990', 'host': 'Player 1'}, {'hostid': '10991', 'host': 'Main
Game'}]
```

Change API object authentication to admin token

```
class Zabbix(AsyncMixin):
  ...
  # ! IMPLEMENT - API login method
  async def apiLogin(self):
    await self.api.login(token=config.zabbixAdminToken)
```

```
$ ./play.py
Hosts: [{'hostid': '10084', 'host': 'Zabbix server'}, {'hostid': '10990 ',
'host': 'Player 1'}, {'hostid': '10991 ', 'host': 'Main Game'}]
```

API respects the user permission scope. **Change token back to player token and remove print(), logout() and exit() code lines.**

play.py

Now the **Zabbix()** class can initialize correctly, however, nothing happens. This is because we need to run some functions that collect data.

Inside **Game()** class **run()** method, we can run function that would request the current player position.

```
class Game(AsyncMixin):
  ...
async def run(self):
    try:
      ...
      # ! IMPLEMENT - launch multiple functions with asyncio
      await self.zabbix.setCurrentPosition()
```

Running this code will fail with exception, because this function does not yet exist in **Zabbix()** class, therefore, we need to add it.

ZABBIX

play.py

This method runs continuously and gets the latest value of position item for player host

```python
class Zabbix(AsyncMixin):
  ...
  # ! IMPLEMENT - request current player position
  async def setCurrentPosition(self) -> list:
    while True:
      currentPosition = await self.api.item.get(
        hostids=self.playerHostId,
        search={'key_': config.playerPositionKey},
        output=['lastvalue']
      )
      print(f'currentPosition: {currentPosition}')
      currentPosition = currentPosition[0]['lastvalue']
      currentPosition = str(currentPosition).split(' ')
      currentPosition[0] = int(currentPosition[0])
      currentPosition[1] = int(currentPosition[1])
      self.playerPosition = currentPosition
      await asyncio.sleep(0)
```

```
$ ./play.py
currentPosition: [{'itemid': '48368', 'lastvalue': '16 11'}]
currentPosition: [{'itemid': '48368', 'lastvalue': '16 11'}]
currentPosition: [{'itemid': '48368', 'lastvalue': '16 11'}]
currentPosition: [{'itemid': '48368', 'lastvalue': '16 11'}]
currentPosition: [{'itemid': '48368', 'lastvalue': '16 11'}]
```

ZABBIX

play.py

We now can successfully call async functions from **Zabbix()** class inside **Zabbix()** class.

This is the **move()** method currently. When called it should print text to console.

```python
class Zabbix(AsyncMixin):
  ...
  async def move(self, direction):
    # Get position
    position = self.zabbix.getCurrentPosition()
    x = position[0]
    y = position[1]

    # Set coordinates for next position
    if direction == 119:
      y = position[1] - 1
    elif direction == 97:
      x = position[0] - 1
    elif direction == 115:
      y = position[1] + 1
    elif direction == 100:
      x = position[0] + 1
    else:
      return

    # ! IMPLEMENT - async Zabbix sender to send updated position values
    # Check if new position is not out of map and does not collide with a wall
    if 0 <= x < config.boardSize and 0 <= y < config.boardSize:
      print(f'move to direction: {chr(direction).upper()}')
      # cellValue = self.gameMap[y][x*2]
      # if cellValue not in config.symbolsWalls:
```

play.py

The **move()** method gets called from **Game()** class method named **movePlayer()**, which monitors for keyboard input from user and then calls **move()**.

```python
class Game(AsyncMixin):
  ...
  async def movePlayer(self):
    while True:
      if self.ui.controlKey != '':
        await self.move(ord(self.ui.controlKey))
        self.ui.controlKey = ''
        await asyncio.sleep(0)
```

However, for debug purposes, in **Game()** there is also a function **movePlayerNoUI()**. We will be using this function until the UI object gets initialized.

```python
class Game(AsyncMixin):
  ...
  async def movePlayerNoUI(self):
    while True:
      await self.zabbix.move(ord(self.getch()))
      await asyncio.sleep(0.5)
```

play.py

Let’s try calling the **movePlayer()** method from **run()**

```python
class Game(AsyncMixin):
  ...
  async def run(self):
    try:
      ...
      # ! IMPLEMENT - launch multiple functions with asyncio
      await self.zabbix.setCurrentPosition()
      await self.movePlayer()
```

```
$ ./play.py
currentPosition: [{'itemid': '48368', 'lastvalue': '16 11'}]
currentPosition: [{'itemid': '48368', 'lastvalue': '16 11'}]
currentPosition: [{'itemid': '48368', 'lastvalue': '16 11'}]
currentPosition: [{'itemid': '48368', 'lastvalue': '16 11'}]
currentPosition: [{'itemid': '48368', 'lastvalue': '16 11'}]
```

Nothing happens differently because **getCurrentPosition()** has not completed its job as it is in an infinite loop. Therefore, **movePlayer()** will never execute.

play.py

This can be fixed with **asyncio.gather()** which will execute both functions concurrently.

```
class Game(AsyncMixin):
  ...
async def run(self):
    try:
      ...
      # ! IMPLEMENT - launch multiple functions with asyncio
      await asyncio.gather(self.zabbix.setCurrentPosition(),
                           self.movePlayerNoUI())
```

movePlayer() relies on **UI()** class for getting non-blocking user inputs, which we have not yet enabled for debugging purposes, therefore, for now we will use **movePlayerNoUI()**

```
$ ./play.py
move to direction: W
currentPosition: [{'itemid': '51772', 'lastvalue': '9 7'}]
currentPosition: [{'itemid': '51772', 'lastvalue': '9 7'}]
currentPosition: [{'itemid': '51772', 'lastvalue': '9 7'}]
move to direction: A
currentPosition: [{'itemid': '51772', 'lastvalue': '9 7'}]
currentPosition: [{'itemid': '51772', 'lastvalue': '9 7'}]
move to direction: S
currentPosition: [{'itemid': '51772', 'lastvalue': '9 7'}]
currentPosition: [{'itemid': '51772', 'lastvalue': '9 7'}]
move to direction: D
currentPosition: [{'itemid': '51772', 'lastvalue': '9 7'}]
currentPosition: [{'itemid': '51772', 'lastvalue': '9 7'}]
```

play.py

To change the player location, we must send new location with Zabbix Sender

First, import AsyncSender from zabbix_utils

```python
from zabbix_utils import AsyncZabbixAPI, AsyncSender
```

Then, we need to create Zabbix sender object in Zabbix class

```python
class Zabbix(AsyncMixin):
    ...
  async def __ainit__(self):
      ...
      # ! IMPLEMENT - asynchronous Zabbix sender object
      self.sender = AsyncSender(server=config.zabbixServerIP,
                                port=config.zabbixServerPort)
```

Now, we can access and use sender from **Game** class **move()** method

ZABBIX

play.py

Implement async sender under first IF statement temporarily. Later we will move it a bit.

```python
class Zabbix(AsyncMixin):
  ...
  async def move(self, direction):
    # Get position
    position = self.zabbix.getCurrentPosition()
    x = position[0]
    y = position[1]

    # Set coordinates for next position
    if direction == 119:
      y = position[1] - 1
    elif direction == 97:
      x = position[0] - 1
    elif direction == 115:
      y = position[1] + 1
    elif direction == 100:
      x = position[0] + 1
    else:
      return

    # ! IMPLEMENT - async Zabbix sender to send updated position values
    # Check if new position is not out of map and does not collide with a wall
    if 0 <= x < config.boardSize and 0 <= y < config.boardSize:
      await self.sender.send_value(self.playerHostName,
                                   config.playerPositionKey,
                                   f'{x} {y}')
      print(f'move to direction: {chr(direction).upper()}')
      # cellValue = self.gameMap[y][x*2]
      # if cellValue not in config.symbolsWalls:
```

play.py

Now the location changes when pressing keys.

```
$ ./play.py
currentPosition: [{'itemid': '51772', 'lastvalue': '2 0'}]
move to direction: S
currentPosition: [{'itemid': '51772', 'lastvalue': '1 0'}]
currentPosition: [{'itemid': '51772', 'lastvalue': '1 0'}]
move to direction: S
currentPosition: [{'itemid': '51772', 'lastvalue': '1 1'}]
currentPosition: [{'itemid': '51772', 'lastvalue': '1 1'}]
move to direction: S
currentPosition: [{'itemid': '51772', 'lastvalue': '2 1'}]
move to direction: S
currentPosition: [{'itemid': '51772', 'lastvalue': '2 2'}]
currentPosition: [{'itemid': '51772', 'lastvalue': '2 2'}]
move to direction: D
currentPosition: [{'itemid': '51772', 'lastvalue': '2 3'}]
```

backend.py

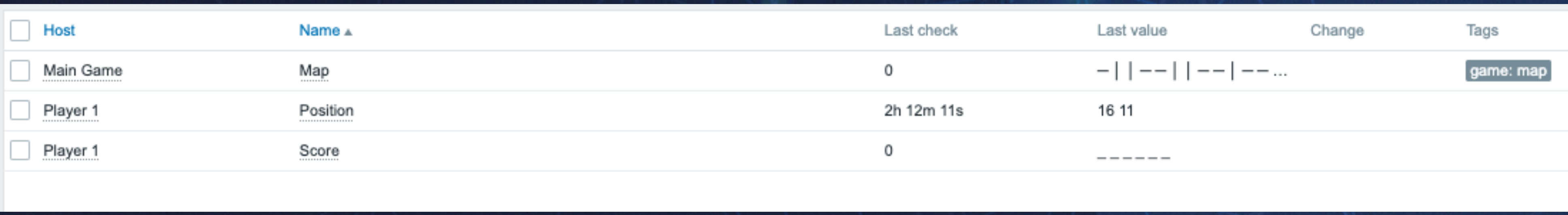
Starting the backend

For Zabbix connectors to be able to send data to our script, we must launch backend.py script which will update game map continuously

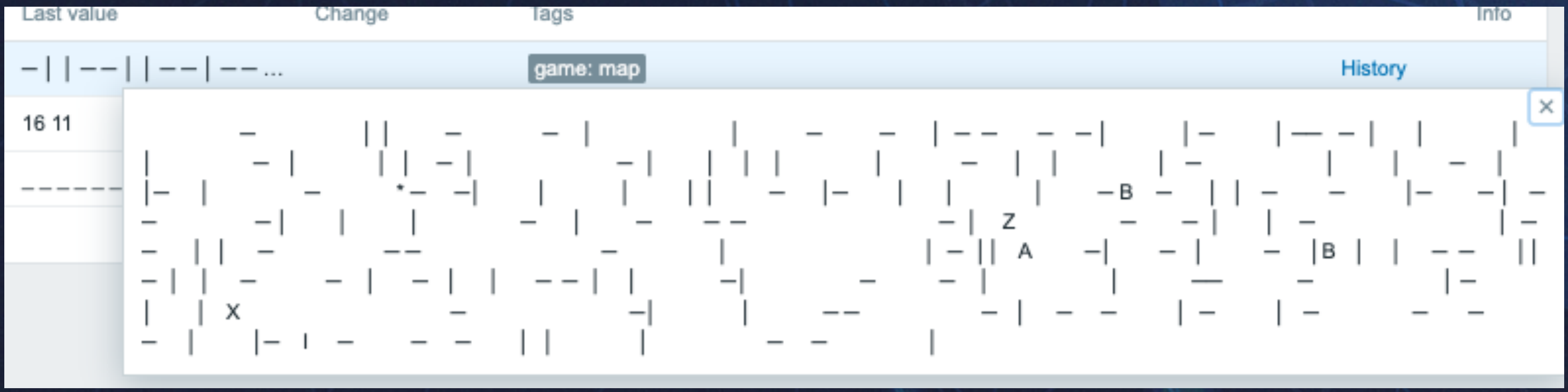
Start the script in a new separate terminal window

```
$ ./backend.py
Backend script running
```

From Zabbix latest data page, Map item on Main Game host is generated and being updated constantly, and Player score item also gets updated



	Host	Name ▴	Last check	Last value	Change	Tags
☐	Main Game	Map	0	−||−−||−−|−−...		game: map
☐	Player 1	Position	2h 12m 11s	16 11		
☐	Player 1	Score	0	------		



Last value	Change	Tags	Info
−||−−||−−|−−...		game: map	History
16 11			

```
      −       ||    −      − |         |     −    −   |−−   − −|     |−    |—−|   |      |
|        − |      || −|          −|   | | |     |      −  | |      |−       |   |   −  |
|−  |      −     *−  −|    |     |   ||    −  |−   |  |      |    −B −  || −   −   |−   −| −
−       −|   |    |      −  |    −    −−                −| Z      −   −|   | −           | −
−   | |  −      −−         −       |             |−|| A    −|   − |    − |B |  |  − −   ||
−|  |  −    − |  − |  |  −−| |     −|      −    − |        |      —    −         |−
|   |X               −         −|      |     −−      − |  −  −    |−    |−       −  −
−  |   |− ı −    −  −   ||     |        −  −      |
```

ZABBIX

play.py

Additionaly, the script also needs to request player’s score from Zabbix.

In Zabbix class, create a function to request the score.

```
class Zabbix(AsyncMixin):
  ...
  # ! IMPLEMENT - request score
  async def setScore(self):
    while True:
      self.score = (await self.api.item.get(
        search={'key_': 'player.score'},
        output=['lastvalue']
      ))[0]['lastvalue']
      print(self.score)
      await asyncio.sleep(0)
```

Then, add **setScore()** to also be executed by **asyncio.gather()**

```
class Game(AsyncMixin):
  ...
async def run(self):
    try:
      ...
      # ! IMPLEMENT - launch multiple functions with asyncio
      await asyncio.gather(self.zabbix.setCurrentPositon(),
                           self.movePlayerNoUI(),
                           self.zabbix.setScore())
```

```
$ ./play.py
currentPosition: [{'itemid': '51772', 'lastvalue': '3 4'}]
Move to direction: S
currentPosition: [{'itemid': '51772', 'lastvalue': '1 0'}]
score:
```

Score currently is empty, but it works and will become occupied when **backend.py** script will be launched

play.py

To enable UI, we need to get game map data. That will be done using Zabbix streaming protocol connectors. This script uses simple HTTP server, to receive data.

Uncomment web server initialization Game class run()

```python
class Game(AsyncMixin):
  ...
  async def run(self):
    try:
    """"""
    threadHTTP = threading.Thread(target=self.startHttpServer)
    threadHTTP.start()

    time.sleep(2)
```

This also allows to complete the **move()** function. Uncomment cellValue and last IF statement and move sender under it

```python
class Zabbix(AsyncMixin):
  ...
  async def move(self, direction):
    ...
    # ! IMPLEMENT - async Zabbix sender to send updated position values
    # Check if new position is not out of map and does not collide with a wall
    if 0 <= x < config.boardSize and 0 <= y < config.boardSize:
      print(f'move to direction: {chr(direction).upper()}')
      cellValue = self.gameMap[y][x*2]
      if cellValue not in config.symbolsWalls:
        await self.zabbix.sender.send_value(self.zabbix.playerHostName,
                                            config.playerPositionKey,
                                            f'{x} {y}')
```

And the script is mostly ready.

play.py

Uncomment UI class initialization

Uncomment everything in **Game** class **run()** method

Change **movePlayerNoUI()** to **movePlayer()**.

Remove print statements we added in functions:
- zabbix.setCurrentPosition()
- zabbix.setScore()
- zabbix.move()

```
class Game(AsyncMixin):
  ...
  async def __ainit__(self):
    self.zabbix = await Zabbix()
    self.ui = UI() # <--- uncomment this line
  ...
  async def run(self):
    try:
    """"""
    threadHTTP = threading.Thread(target=self.startHttpServer)
    threadHTTP.start()

    time.sleep(2)

    threadSetMap = threading.Thread(target=self.zabbix.setMap)
    threadSetMap.start()

    threadUiUpdate = threading.Thread(target=self.updateScreen)
    threadUiUpdate.start()

    threadSync = threading.Thread(target=self.synchronizeData)
    threadSync.start()

    threadInput = threading.Thread(target=self.askInput)
    threadInput.start()

    # ! IMPLEMENT - launch multiple functions with asyncio
    await asyncio.gather(self.zabbix.setCurrentPositon(), self.movePlayer(),
                         self.zabbix.setScore())
```

play.py

Fixing Game() class move() method

And now you should see that also score is being printed to the terminal

```
$ ./play.py

           Z A B B I X
          ┌──────────────────────┐
          │                      │
          │      |               │
          │              |     | │
          │                      │
          │        |             │
          │_         *           │
          │            _         │
          │  |         _   X     │
          │                      │
          │_       _             │
          │                      │
          └──────────────────────┘
```

ZABBIX

Thank you!

Kristaps Naglis

Integration Engineer