سند بررسی برنامه **Pinger** با زبان پایتون

ویرایش نخست

نویسنده این سند: علیرضا سلطانی نشان

۱۴۰۰,۰۹,۰۷

هدف برنامه:

استفاده در سیستم های کلاینت (به عنوان سرویس دهنده) در پروژه پارکینگ هوشمند **همراه اول**

فهرست سند

مقدمه:

برنامه Pinger یک برنامه ساده برای بررسی برقرار ارتباط با نود های یک شبکه است. مثلا برای اینکه بدانیم سیستم n می‌تواند دوربین های مورد نظر را در شبکه ببیند با استفاده از برقرار یک HS[1] از صحت ارتباط بین نود ها اطمینان حاصل می کند.

زبان و کلیات:

با استفاده از زبان python نوشته شده است. حاوی ارتباط با دیتابیس MongoDB است. با استفاده از PDM[2] مدیریت وابستگی های برنامه قرار می‌گیرد.

بسته های استفاده شده:

```
    "pymongo~=3.12",
    "requests~=2.18",
```

دلیل استفاده از این دو بسته:

استفاده از وابستگی pymongo برای ارتباط با دیتابیس به صورت مستقیم.

استفاده از وابستگی requests برای request handling بین اسریکپت پایتون و ممکن است با API برنامه Laravel.

برنامه چگونه کار می‌کند:

برنامه با استفاده از یک فایل پیکربندی مرتبط با الگوی که داده شده (که در ادامه می‌نگارم) عمل می‌کند. پس از دریافت فایل پیکربندی، روی هر نود کافی است با استفاده از دستور زیر در ps[3] ویندوز مورد اجرا قرار بگیرد:

[1] Hand Shaking
[2] Python Dependencies Manger
[3] Power Shell

- **In Unix (nix) system:**

python or python3 main.py –b building_name –i current device IP Address

- **In Windows or Power shell:**

Py or python main.py –b building_name –i current device IP Address

پارامتر های ورودی

اجرای دستور به صورت زیر است:

- **نام ساختمان یا building_name :**

Python –building_name or –b

- **آدرس IP دستگاهی که این اسکریپت را اجرا می‌کند:**

Python –ip or –I

نکته: مقادیری که برای آپشن b– درنظر گرفته می‌شود باید مانند نام متغیر هایی باشد که در فایل پیکربندی نوشته شده است.

- **نمونه حقیقی استفاده از اسکریپت:**

Python main.py –b setare –i 10.99.176.60

فایل پیکربندی:

```
time_to_wait = 100

main_config = {
    "db": {
        # Loopback DB
        "conn_string":
"mongodb://localhost:27017/?readPreference=primary&appna
me=MongoDB%20Compass&ssl=false",
        "db_name": "CPR_Paya_DB",
        "collection_name": "Syslog",
    }
}

setare = {
    "server": [
        {"server1": "10.99.176.60"},
        {"server2": "10.99.176.61"},
        {"server3": "10.99.176.62"},
    ],
    "camera": [
        {"camera1": "10.99.176.50"},
        {"camera2": "10.99.176.51"},
        {"camera3": "10.99.176.52"},
        {"camera4": "10.99.176.53"},
    ],
    "gateway": [
        {"gateway1": "10.99.176.113"},
        {"gateway2": "10.99.176.114"},
        {"gateway3": "10.99.176.115"},
        {"gateway4": "10.99.176.116"},
        {"gateway5": "10.99.176.117"},
        {"gateway6": "10.99.176.118"},
        {"gateway7": "10.99.176.119"},
    ],
}

# Check and set new values for main building IP.
main = {
    "server": [
        {"server1": "10.99.176.60"},
        {"server2": "10.99.176.61"},
```

```
        {"server3": "10.99.176.62"},
    ],
    "camera": [
        {"camera1": "10.99.176.50"},
        {"camera2": "10.99.176.51"},
        {"camera3": "10.99.176.52"},
        {"camera4": "10.99.176.53"},
    ],
    "gateway": [
        {"gateway1": "10.99.176.54"},
        {"gateway2": "10.99.176.55"},
        {"gateway3": "10.99.176.56"},
        {"gateway4": "10.99.176.57"},
        {"gateway5": "10.99.176.58"},
    ],
}

huawei = {
    "server": [
        {"server1": "10.90.0.13"},
        {"server2": "10.90.0.14"},
        {"server3": "10.90.0.15"},
    ],
    "camera": [
        {"camera1": "10.90.0.17"},
        {"camera2": "10.90.0.18"},
        {"camera3": "10.90.0.19"},
        {"camera4": "10.90.0.20"},
    ],
    "gateway": [
        {"gateway1": "10.90.0.21"},
        {"gateway2": "10.90.0.22"},
        {"gateway3": "10.90.0.23"},
        {"gateway4": "10.90.0.24"},
        {"gateway5": "10.90.0.25"},
        {"gateway6": "10.90.0.25"},
    ],
}
```

برای استفاده از برنامه بایستی مانند الگوی بالا پیش بروید.

نکته: با تغییر همین قسمت که بیشتر وابستگی برنامه را نشان می‌دهند می‌توان در هر محیطی از آن استفاده کرد.

در ادامه عملکرد برنامه می‌توان اشاره کرد که نحوه بررسی نود ها با استفاده از تابع ps_command است.

این تابع یک آرگومان دارد آن هم آدرس IP نود مقصد است.

```
    try:
        ps_default_path = (
            "C:\\Windows\\System32\\WindowsPowerShell\\v
1.0\\powershell.exe"
        )
        run = [ps_default_path, f"ping -n 1 {ip} |
select-string -pattern 'Reply'"]
        # Decode the output of the command to normal and
simple string witout \n\r and etc...
        res =
str(subprocess.check_output(run).decode("utf-8"))
        # Will return 128 as ttl and keep that as
string, to
        # getting only number of ttl like 64 or 128
        ttl = int(res.split(" ")[5].strip()[4:])  # make
it integer
        # Check if ttl is not 0ms or above of 10ms, True
is okay, and False is not connected.
        if ttl > 10:
            return True
        else:
            return False

    except Exception as err:
        print(f"Error in ping function: {err}")
        return False
```

برنامه ping را در power shell اجرا کرده و فقط یک مقدار از Ping را دریافت می‌کند. یعنی در حالت عادی وقتی شما نود مقصد را پینگ می‌گیرید ۴ بسته ارسال می‌کند به وسیله پروتکل ICMP ولی زمانی که از ping –n 1 استفاده میکنیم فقط یک

بسته ارسال و سپس با pipe قسمت ‘Reply’ select-string -pattern فقط قسمت reply بررسی بسته را انتخاب می‌کند. که مقداری عددی بایستی باشد. درصورتی که این مقدار برای مثال unreachable باشد مقدار False را این تابع بر می‌گرداند.

نکته: لازم به ذکر است، باتوجه به نام ساختمان وارد شده در قسمت دستور شل، بعد از اینکه لیست نود های مورد نظر با دستور پینگ آزمایش شد برای امتحان روی مقدار ۱۰۰ ثانیه صبر می‌کند و دوباره این عملیات را تکرار می‌کند. در صورت کم یا زیاد کردن این مقدار می توانید با اضافه کردن مقدار مناسب در فایل کانفیگ اقدام کنید.

```
time_to_wait = 10 # or anything
```

ساختار داده ذخیره شده در DB:

```
def insert_one(self, systemLog: Dict):
        """Insert_one from the important section.
        That will get system log dictionary and will set
in proper collection.
        Dictionary of entry:
        {
            src: string,
            dst: string,
            node: string,
            timestamp: int
            dateTime: string
        }
        """
        self.collection.insert_one(systemLog)
```