Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Správa webserveru

Zátěžové testování
Laboratorní úloha

Ing. Lukáš Čegan, Ph.D.

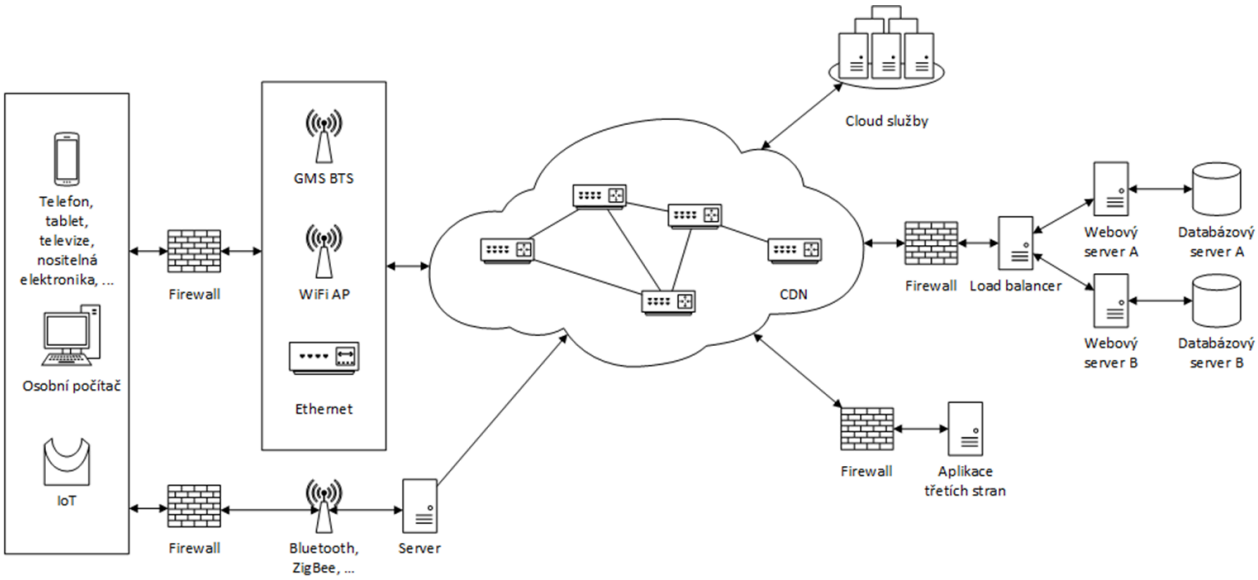
Správa webserveru - laboratorní úloha

Teoretická východiska laboratorní úlohy

Testování výkonu webových aplikací a webových serverů

Testování výkonu je nedílnou součástí zajišťování kvality provozu webových aplikací a příslušné infrastruktury. Na jedné straně testujeme platformu, na které provozujeme webovou aplikaci a na druhé straně testujeme samotnou webovou aplikaci.

Testování výkonu je způsob zkoumání a hodnocení chování systému. Během testování výkonu měříme odezvu systému, rychlost, stabilitu, vyhodnocujeme spolehlivost a měříme využití systémových zdrojů infrastruktury. Je nutné si uvědomit, že na komunikační cestě mezi klienty (konzumentem služby) a serverem (poskytovatelem služby) se vyskytuje řada aktivních prvků, jejichž výkon se přímo promítá do výsledné kvality služby a v řadě případů není možné přímo ovlivnit chování/konfiguraci těchto prvků (viz obrázek níže).



Zdroj: vlastní

Před samotným testováním výkonu je nutné explicitně vyjádřit očekávané chování sytému, který podrobujeme testu, aby bylo možné jednoznačně vyhodnotit míru shody chování testovaného systému s jeho očekávaným chováním.

Jako hlavní nefunkční požadavky, které jsou předmětem testování, můžeme zařadit:

- počet paralelních uživatelů,
- počet přístupů za sekundu,
- využití paměti pro zpracování,
- počet chyb (např. stránek) za sekundu,
- průměrná doba odezvy,
- velikost použité operační paměti,
- doba zaneprázdnění disku (I/O operace),
- latence na síti,
- šířka pásma sítě (počet bitů za sekundu),
- počet SQL dotazů uložených v mezipaměti,
- maximální počet relací, které mohou být současně aktivní,
- maximální doba čekání na zpracování,
- rychlost, kterou se nevyužitá paměť vrací do systému (Garbage collection).

Při identifikace úzkých míst a chyb v testovaném systému je nutné zvolit vhodné nápravné opatření, kterým bude možné přiměřeně dosáhnout předem stanovených byznys cílů (v souladu se stanovenými KPI).

Předmětem testování se stávají jak samostatné komponenty systému, tak i systém jako celek. Samostatně se testuje funkčnost:

- webových serverů (včetně proxy serverů),
- databázových serverů,
- síťové infrastruktury a připojení,
- diskových polí,
- komponenty třetích stran
- a podobně (v závislosti na celkové architektuře řešení).

Testování výkonu zahrnuje řadu testů, které rozlišujeme podle situace, na kterou test cílíme. Mezi ty nejznámější zahrnujeme:

- zátěžové testy,
- stres testy,
- objemové testy,
- soak testy,
- regresní testy,
- spike testy,
- testy škálovatelnosti,
- testy odolnosti.

Zátěžové testování

Testování zátěže je typ testu výkonu, kterým testujeme fungování sytému při předpokládaném zatížení. Test provádíme s definovaným počtem souběžných virtuálních uživatelů, kteří provádějící typické transakce za určité časové období. Jinými slovy, zátěžové testování měří, jak systémy zvládají očekávané objemy zátěže.

Stres test

Stresový test je typ testu výkonu, který testuje horní limity systému při extrémní zátěži. Jedná se o test hraničních hodnot. Tímto testem zjišťujeme chová systému při intenzivní zátěži a jeho chování při návratu k normálnímu provozu (tzv. zotavení systému). Jako klíčové ukazatele u tohoto typu testu můžeme považovat "propustnost" a "dobu odezvy". Stres test byl nám měl být schopen odhalit hranici pro odmítnutí služby, neúměrné zpomalení odezvy, výskyt bezpečnostních problémů či poškození dat. Tento typ testu se provádí se často provádí v distribuovaném prostředí za pomocí nástrojů umožňující velmi vysokým počtem souběžných virtuálních uživatelů.

Spike test

Podstatou Spike testu je sledování chování výkonu systému při rychlém zvýše počtu požadavků až na úroveň stresu a jeho opětovné rychlé snížení. Tento test se následně opakuje v náhodných nebo konstantních intervalech, aby bylo možné sledovat chování systému v dlouhé periodě. Tento typ testu se hodí pro testování špiček, jako jsou například výprodejové akce eshopů a podobně.

Objemové testy

Objemové testy se provádí pro sledování chování systému, který zpracovává velký objem dat nebo dopad postupně se zvyšujícího se objemu dat na chování systému.

Testy odolnosti

Prostřednictvím testů odolnosti sledujeme stabilitu a výkon systému v dlouhém časovém okně při stálém pracovním zatížení. Cílem testu je odhalit snížení výkonu, pokles disponibilní paměti, nedostupnost vláken a další problémy, které mohou nastat při dlouhém nepřetržitém běhu aplikace(dny, týdny).

Regresní testy

Regresní testy se provádějí z důvodu odhalení negativních dopadů na výkon systému z důvodu zavádění změn v kódu aplikace. Pomocí těchto testů zjišťujeme, zda nové funkce či opravy nezpůsobovaly regresi výkonu nebo nesnižovaly celkový výkon systému ve srovnání s předchozím stavem.

Testy škálovatelnosti

Testy škálovatelnosti se provádějí pro potřebu měření chování systému při změně vybraných atributů systému (nahoru, dolu) s cílem zjistit, zda zvýšení/snížení těchto atributů použitého systému je přímo úměrné očekávanému výkonu aplikace (zda faktor škálovatelnosti je blízko násobku zatížení).

Proces testování

Testování je nedílnou součástí procesu vývoje webových aplikací. V rámci tohoto procesu postupujeme v následujících krocích:

1. Stanovení testovacích metrik - metriky by měly být odvozeny ze základních požadavků na software a dále by měly reflektovat stanovené obchodní cíle
2. Stanovení testovacích scénářů - explicitní vyjádření, jak bude testování probíhat (krok za krokem) a z jakými daty.
3. Stanovení testovací platformy - v jakém prostředí s jakými nástroji budou probíhat testy (např. JMeter, Taurus, Gatling, ...)

4. Sestavení testovacího skriptu - skript, který je vytvořena na základě stanoveného testovacího scénáře a který je možný opakovaně spustit na testovací platformě.
5. Provedené testu - spuštění testovacího skriptu, aktivní monitoring probíhajícího testu a jeho logování.
6. Analýza výsledků testu - investigace výsledků testu, identifikace případných úzkých místa a chyb systémů.
7. Vytvoření a zavedení nápravných opatření a opětovné spuštění testu (prováděno v cyklu do té doby, než systém splňuje nastavené KPI).

Testování je náročný proces, který je nutný spouštět v rámci každého releasu systému. Jinak řečeno, pokud chceme provést změny na produkčním prostředí, musíme předtím vše řádně otestovat. Z tohoto důvodu je nutné testování automatizovat a zahrnout do vývojového procesu v rámci CI/CD.

◄ Krycí list - zadní stana

Přejít na...

Laboratorní úloha - tutoriál ►

Nápověda a dokumentace (anglicky)
Kontaktujte podporu stránek

Jste přihlášeni jako Lukáš Čegan (Odhlásit se)
DANTE_SW_L_FINAL

Moje stránka
Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)
Čeština (cs)
English (en)

Souhrn uchovávaných dat

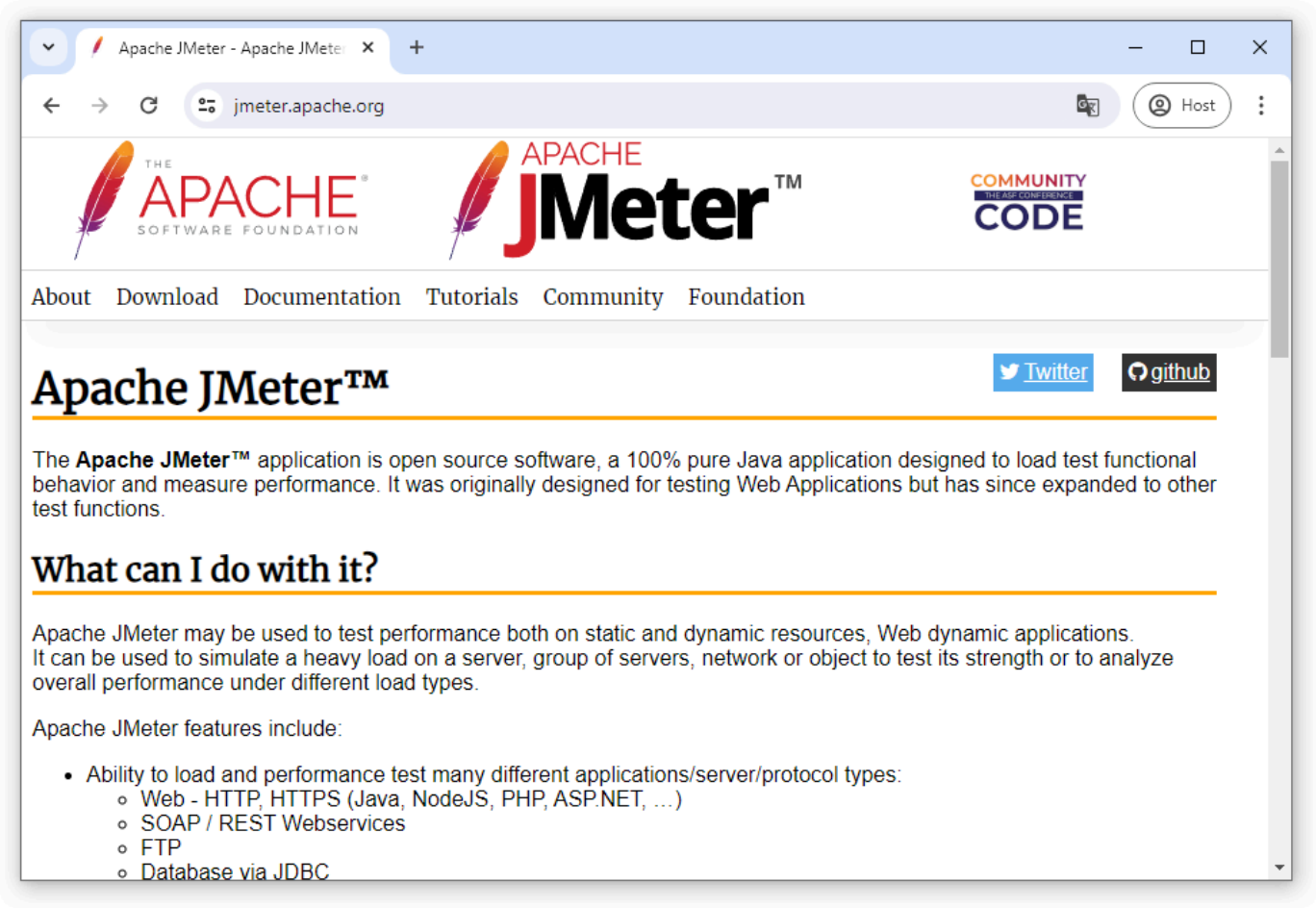
Stáhněte si mobilní aplikaci

Správa webserveru - laboratorní úloha

Laboratorní úloha - tutoriál

Instalace a spuštění aplikace JMeter

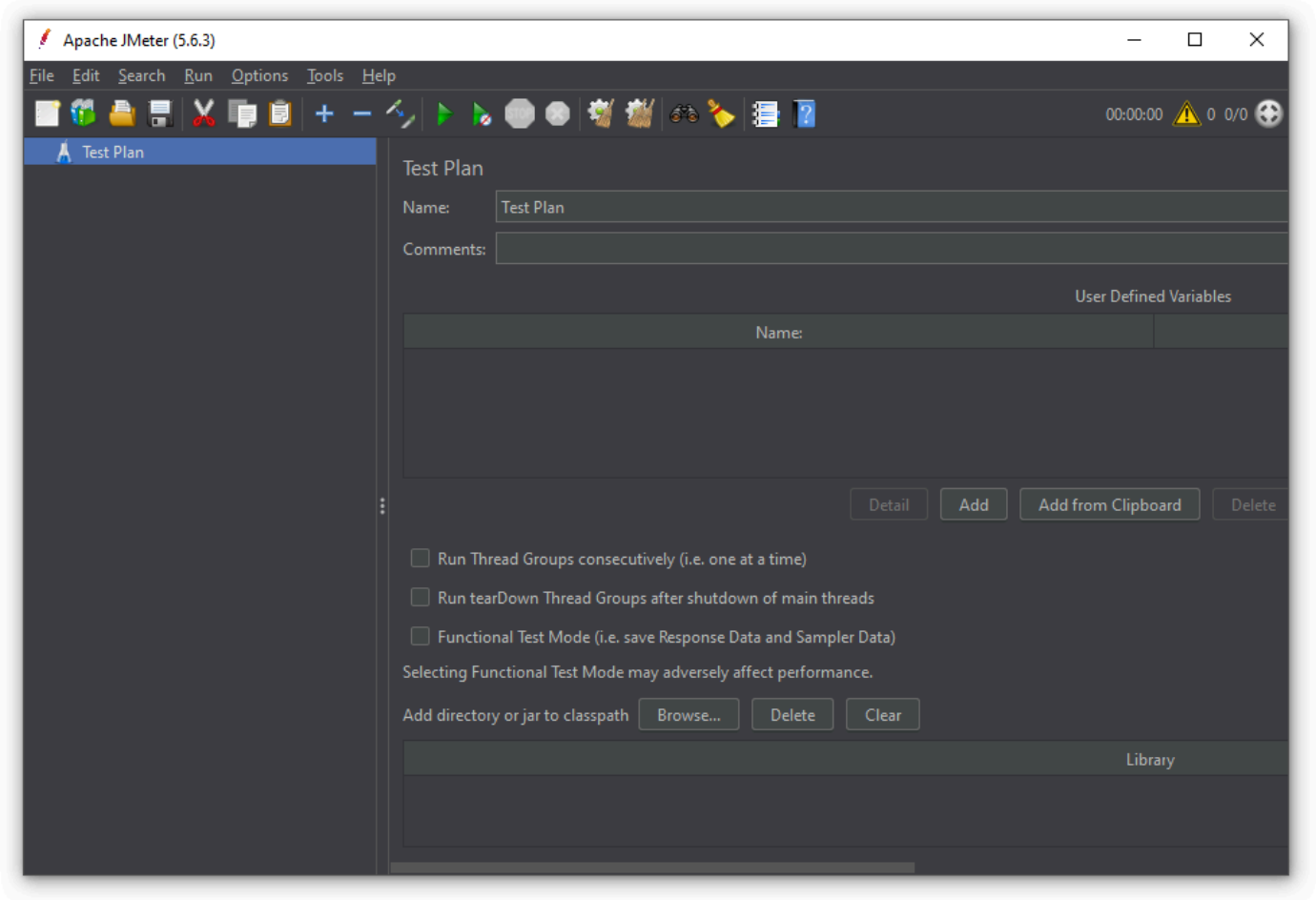
1. Stáhněte aplikaci JMeter z https://jmeter.apache.org/download_jmeter.cgi. Pro spuštění je vyžadována Java 8. V případě, že ji nemáte instalovánu v počítači, stáhněte si ji z https://www.oracle.com/cz/java/technologies/downloads/.



Zdroj: Screenshot jmeter.apache.org

2. Stažený archiv aplikace JMeter extrahujeme do námi zvoleného adresáře, pro který dále v textu budeme používat označení {jmeter_root}.

3. Aplikaci spustíme dávkou {jmeter_root}/bin/jmeter.bat. Tato dávka spustí JMeter v režimu GUI, který by měl být použit pouze pro vytvoření skriptu zátěžového testu. Pro samotné testování zátěže by měl být JMeter spouštěn v režimu CLI, tedy bez GUI.



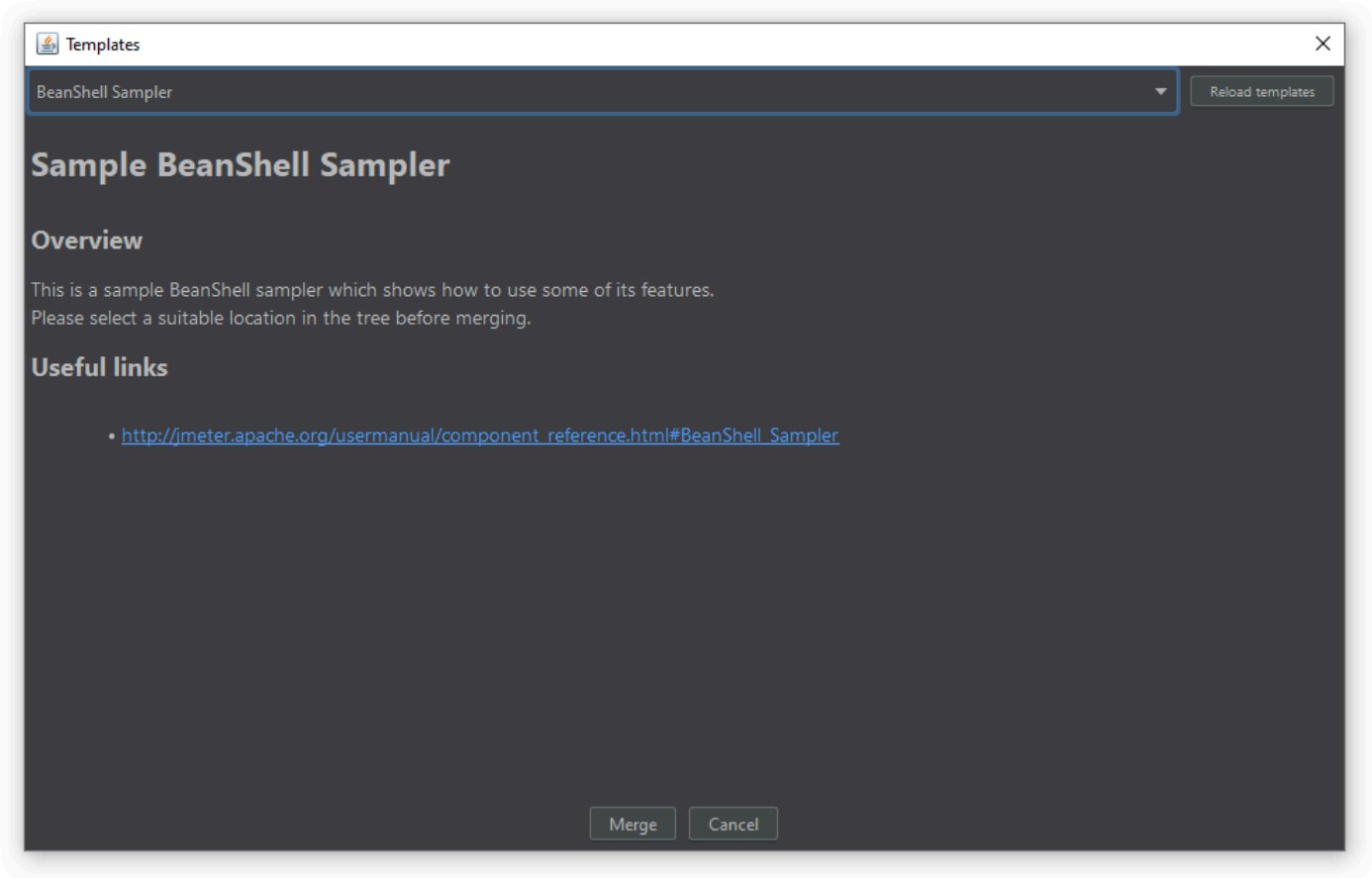
Zdroj: Screenshot aplikace JMeter

Přehled dávek pro práci s aplikací JMeter:

- jmeter.bat - spustí JMeter ve výchozím nastavení v režimu GUI
- jmeterw.cmd - spustí JMeter bez konzoly Windows ve výchozím nastavení v režimu GUI
- jmeter-n.cmd - spustí zátěžový test v režimu CLI se souborem JMX
- jmeter-n-r.cmd - načte soubor JMX pro vzdáleně spuštění zátěžového testu v režimu CLI
- jmeter-t.cmd - spustí JMeter v režimu GUI a načte soubor JMX
- jmeter-server.bat- spustí JMeter v režimu server
- mirror-server.cmd - spouští JMeter Mirror Server v režimu CLI
- shutdown.cmd - spustí klienta pro řádné zastavení instance v režimu CLI
- stoptest.cmd - spustí klienta pro náhlé zastavení běžící instance v režimu CLI

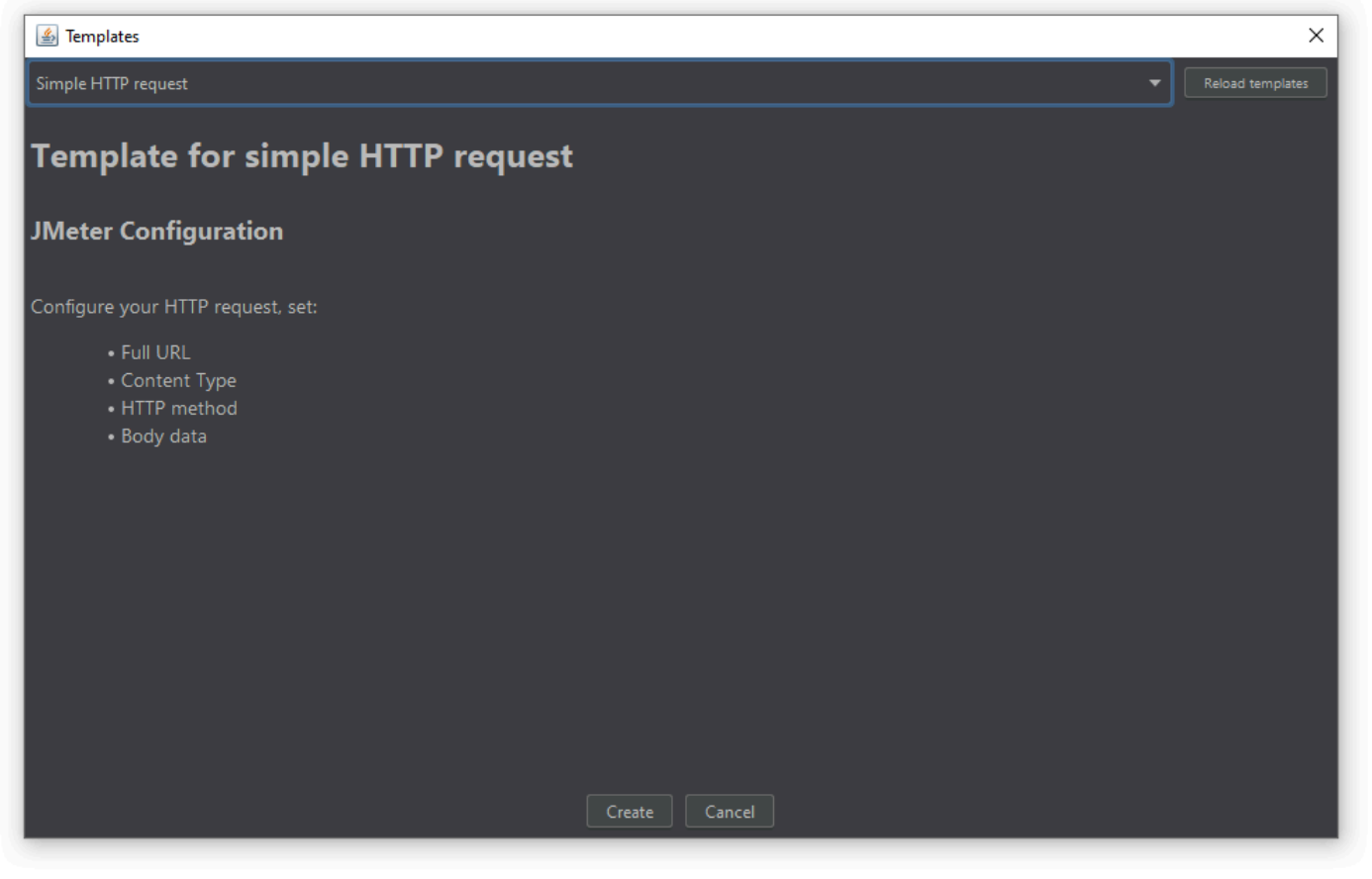
Příprava testovacího plánu ze šablony

1. Vytvořit nový testovací plán z existující šablony lze pomocí nabídky **File** → **Templates**. Zobrazí se nové okno, ve kterém vybereme šablonu testu.



Zdroj: Screenshot aplikace JMeter

2. Zvolíme šablonu **Simple HTTP request** a potvrdíme volbu tlačítkem **Create**.

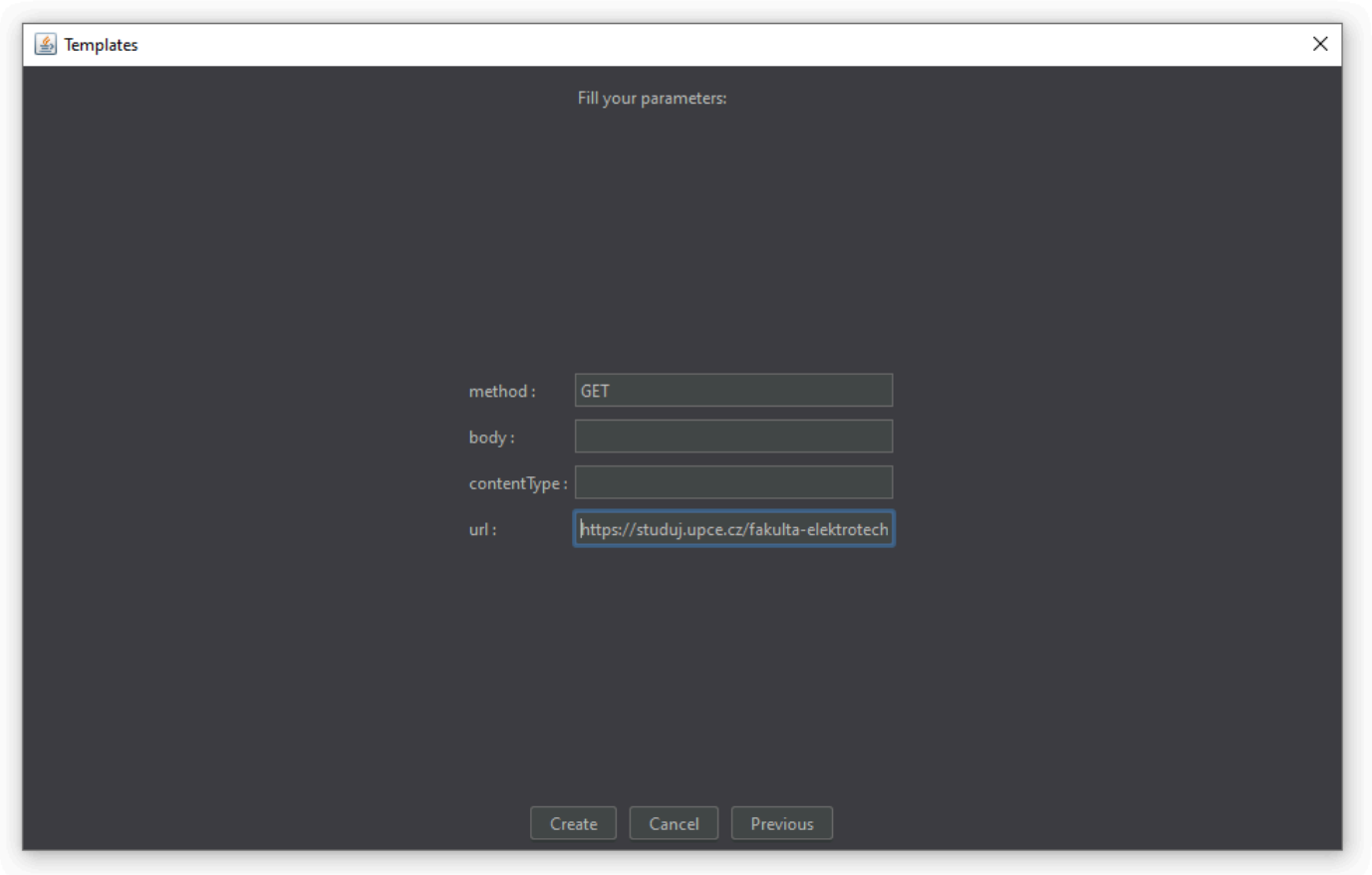


Zdroj: Screenshot aplikace JMeter

3. Na další obrazovce šablony zadáme vstupní parametry testu:

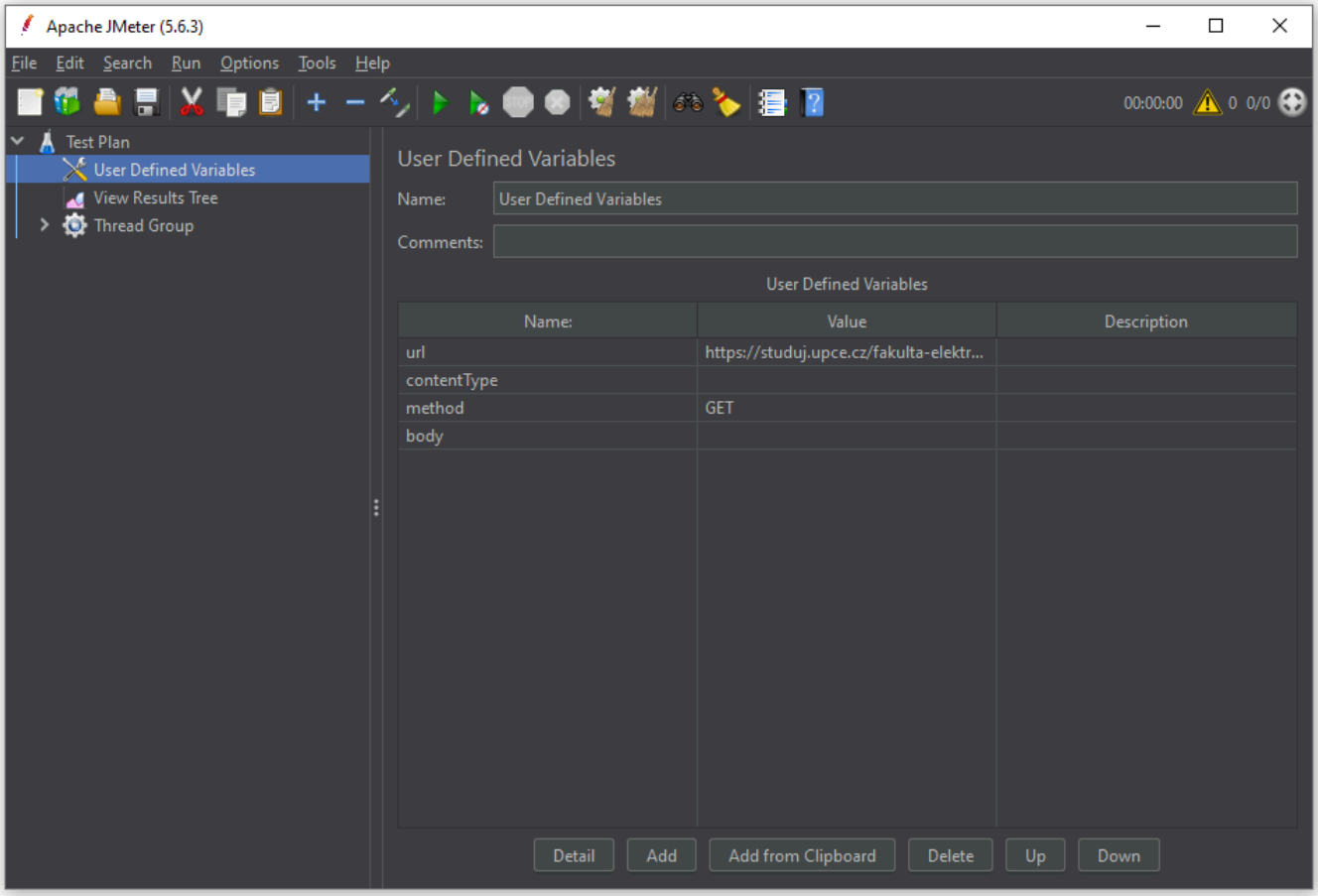
- method: GET
- body:
- contentType:
- url: https://studuj.upce.cz/fakulta-elektrotechniky-informatiky

Zadané parametry potvrdíme tlačítkem **Create**.



Zdroj: Screenshot aplikace JMeter

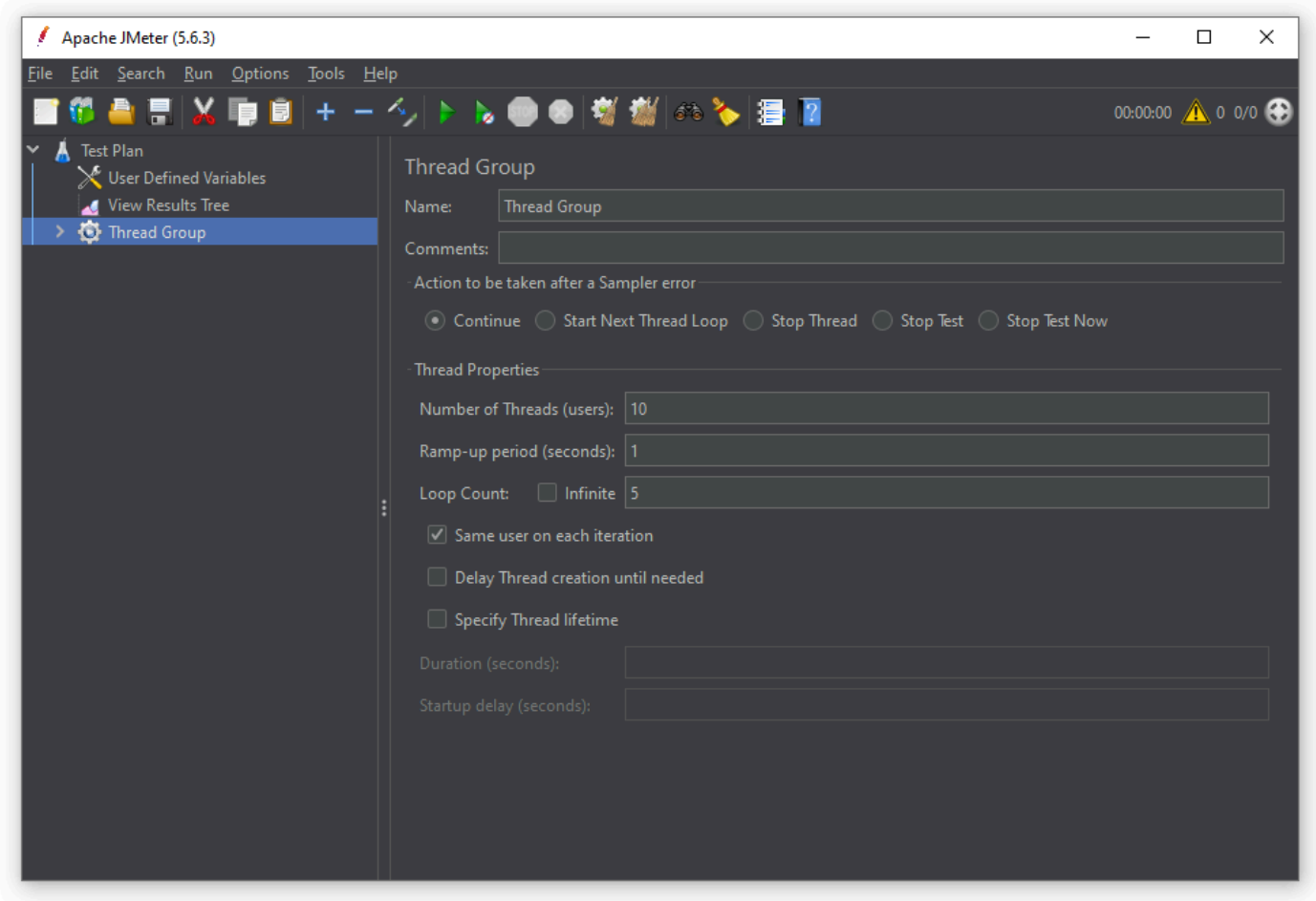
4. Konfigurace zátěžového testu ze šablony se nám vygenerovala a je dostupná v položce testovacího plánu **User Defined Variables**.



Zdroj: Screenshot aplikace JMeter

5. V menu testovacího plánu vybereme položku **Thread Group** a změníme parametry testu:

- Number of Threads (users): 10
- Loop Count: 5



Zdroj: Screenshot aplikace JMeter

6. Vytvořený test uložíme **File** → **Save** do souboru **{jmeter_root}/testy/test_studuj_fei.jmx**.

7. Spuštění zátěžového testu provedeme v režimu CLI, tedy bez GUI. Pro spuštění lze použít následující parametry:

- -n spouští JMeter v režimu cli
- -t [souboru JMX] souboru JMX specifikuje lán testování
- -l [souboru JTL] soubor JTL specifikuje protokolovací soubor s ukázkovými výsledky testu
- -j [souboru] soubor protokolu běhu testu
- -r spustí test na serverech, které jsou specifikovány v vlastnosti "remote_hosts"
- -R [seznam serverů] spustí test na vzdálených serverech
- -g [souboru CSV] soubor CSV specifikuje soubor pro generování řídicího panelu sestav
- -e vygeneruje řídicí panel sestav po zátěžovém testu
- -o výstupní složka, do které se vygenerují řídicí panely sestav po zátěžovém testu
- -H [název hostitele serveru proxy nebo adresa IP]
- -P [port proxy serveru

8. První zátěžový test spustíme pomocí příkazu **{jmeter_root}/bin/jmeter -n -t ../testy/test_studuj_fei.jmx -l ../testy/logy/test_studuj_fei_log_1.jtl**

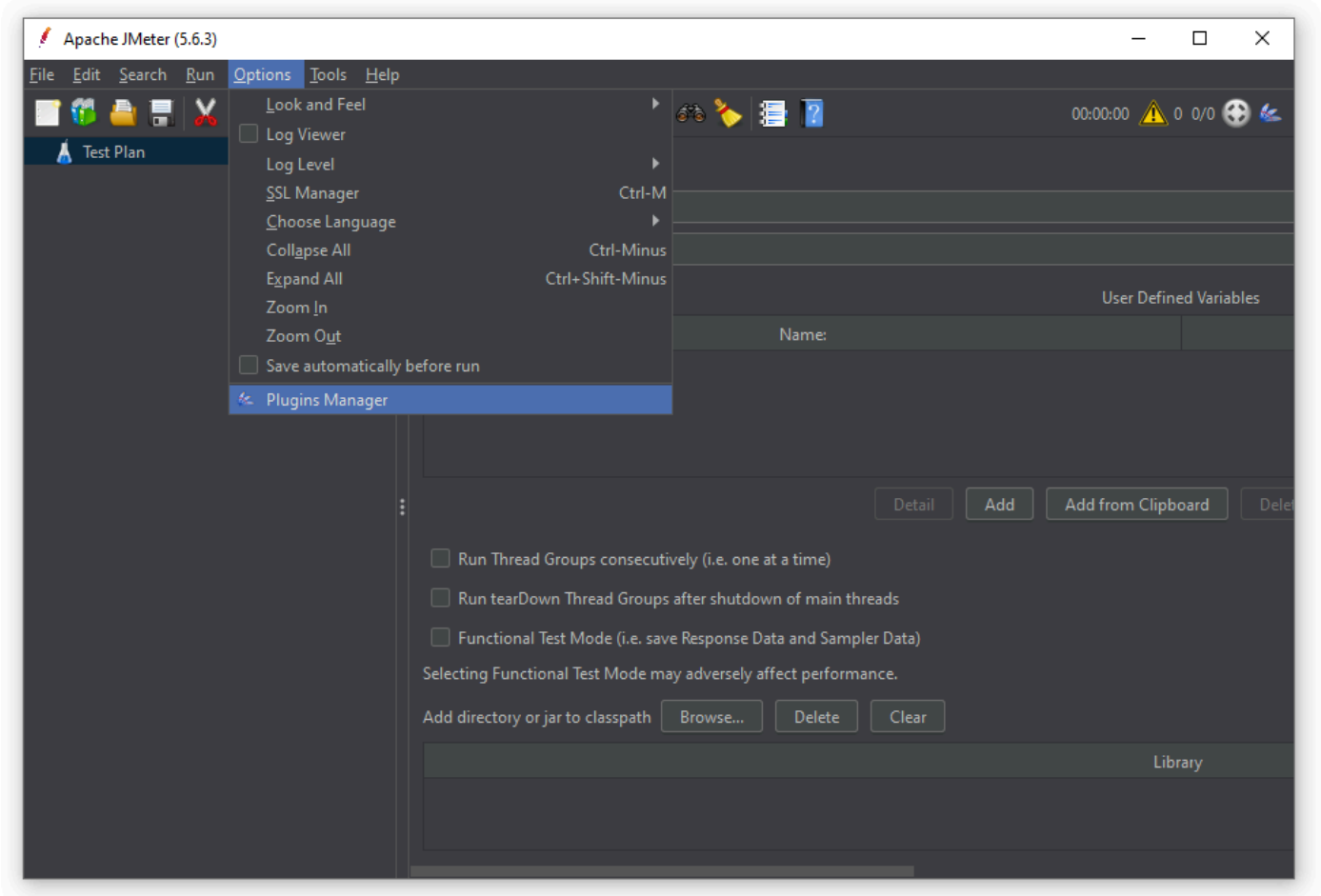
9. Výsledek testu si můžeme prohlédnout v souboru **{jmeter_root}/testy/logy/test_studuj_fei_log_1.jtl**

timeStamp,elapsed,label,responseCode,responseMessage,threadName,dataType,success,failureMessage,bytes,sentBytes,grpThreads,allThreads,URL,Late
1716539335753,63,HTTP Request,200,OK,Thread Group 1-6,text,true,,77109,160,6,6,https://studuj.upce.cz/fakulta-elektrotechniky-informatiky,45,0,25
1716539335370,444,HTTP Request,200,OK,Thread Group 1-2,text,true,,77109,160,6,6,https://studuj.upce.cz/fakulta-elektrotechniky-informatiky,429,0,401
1716539335545,270,HTTP Request,200,OK,Thread Group 1-4,text,true,,77109,160,6,6,https://studuj.upce.cz/fakulta-elektrotechniky-informatiky,253,0,225
1716539335449,366,HTTP Request,200,OK,Thread Group 1-3,text,true,,77109,160,6,6,https://studuj.upce.cz/fakulta-elektrotechniky-informatiky,349,0,321
...

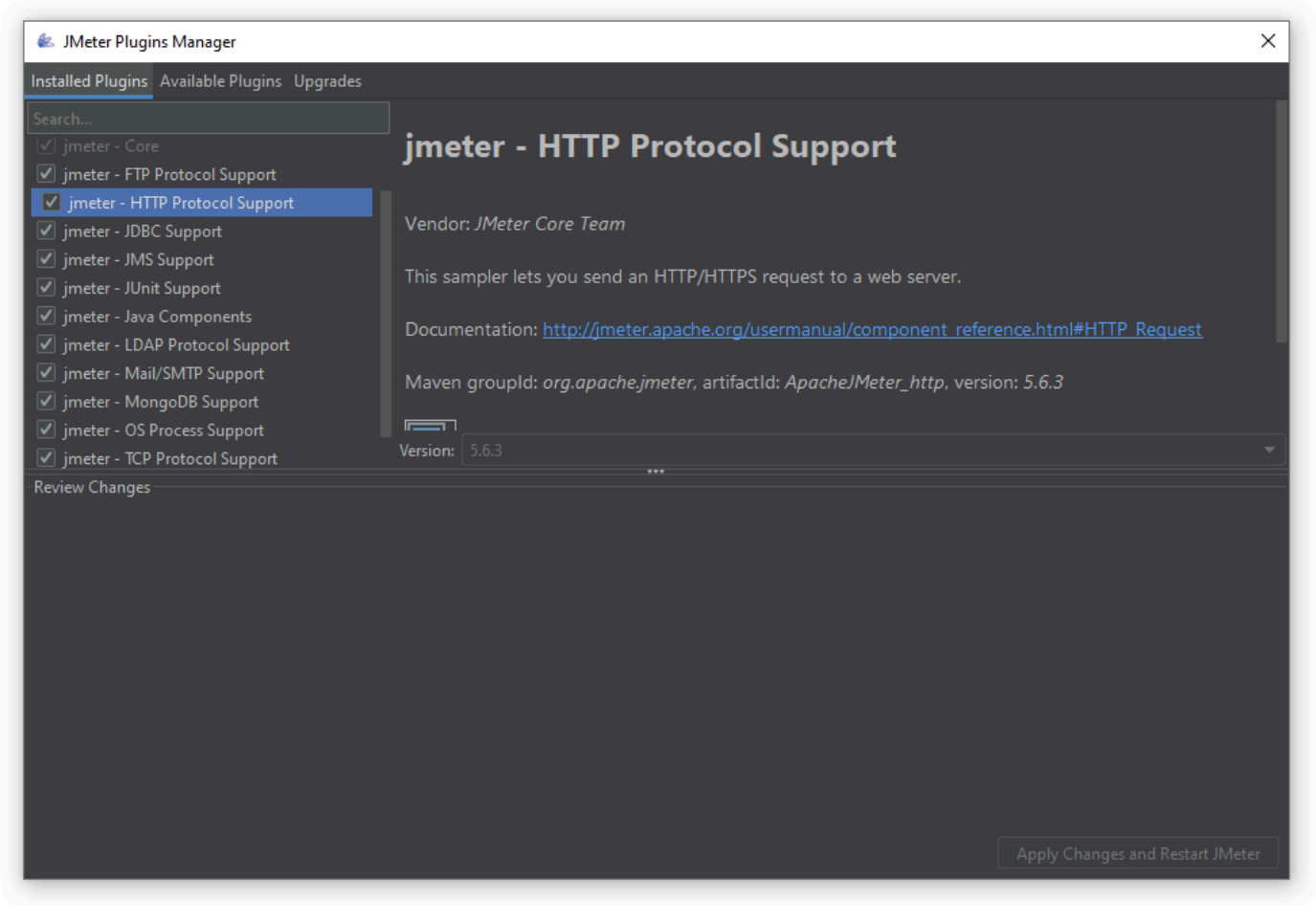
Instalace plugin manageru do JMeter

1. Plugin manager si stáhneme z https://jmeter-plugins.org/install/Install/ a soubor **plugins-manager.jar** umístíme do adresáře **lib/ext**.

2. Restartujeme JMeter a spustíme Plugins Manager (Options → Plugins Manager)



Zdroj: Screenshot aplikace JMeter



Zdroj: Screenshot aplikace JMeter

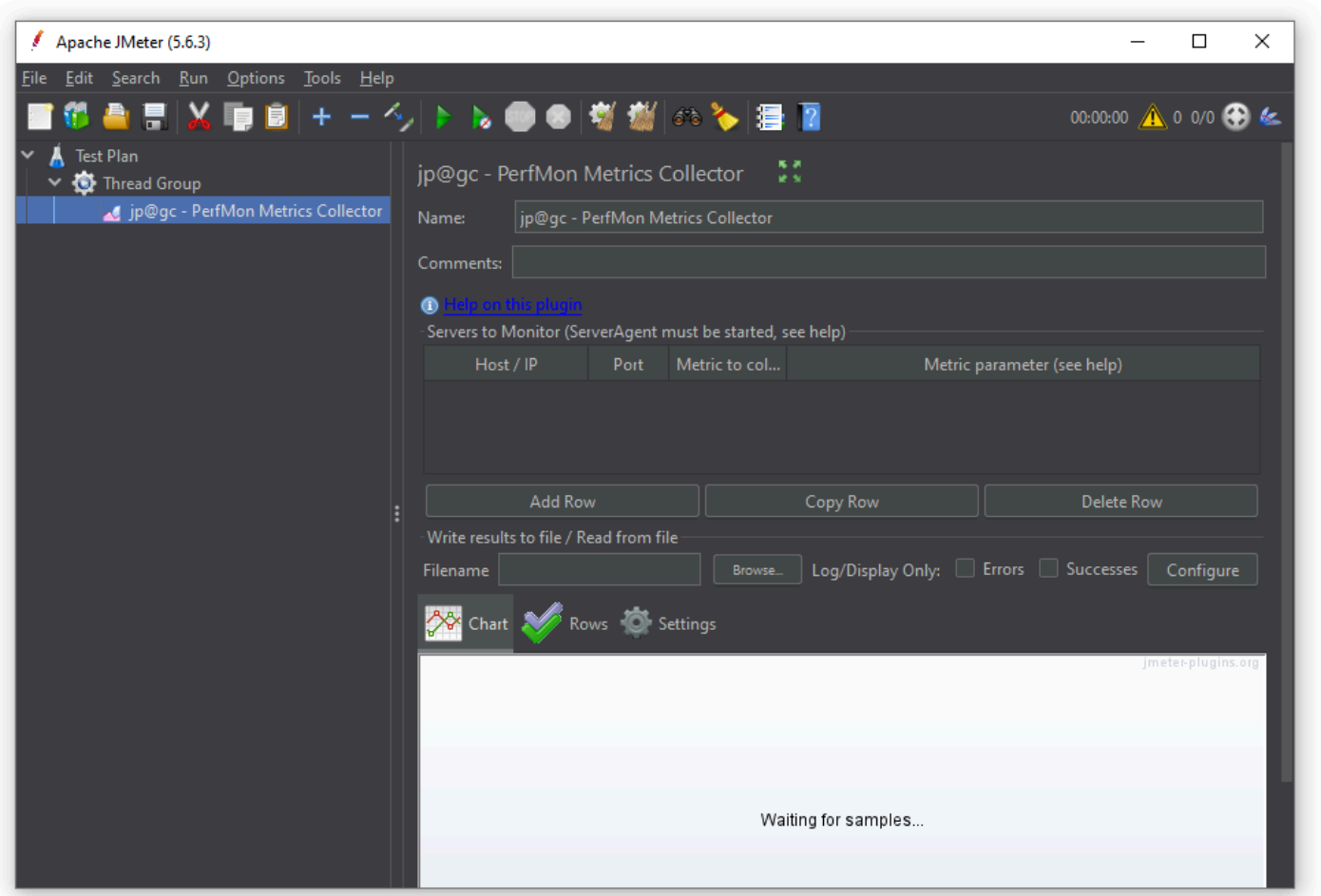
Instalace pluginu PerfMon (Servers Performance Monitoring)

Plugin PerfMon je skvělí nástroj pro analýzu používání CPU, paměti, diskových I/O operací.

1. Plugin manager je dostupný v menu JMeter **Options → Plugins Manager**

2. Na kartě **Available Plugins** do pole pro vyhledávání zadáme **PerfMon** a plugin přidáme do JMeter pomocí tlačítka **Apply changes and Restart JMeter**.

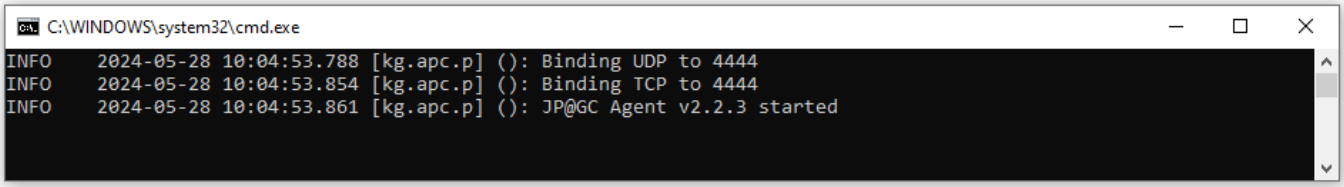
3. Vytvoříme nový TestPlan, do kterého přidáme Thread Group (Add → Threads(Users) → Thread Group)
4. Pro Thread Group přidáme posluchače jp@gc - PerfMon Metrics Collector (Add → Listener(Users) → jp@gc - PerfMon Metrics Collector)



Zdroj: Screenshot aplikace JMeter

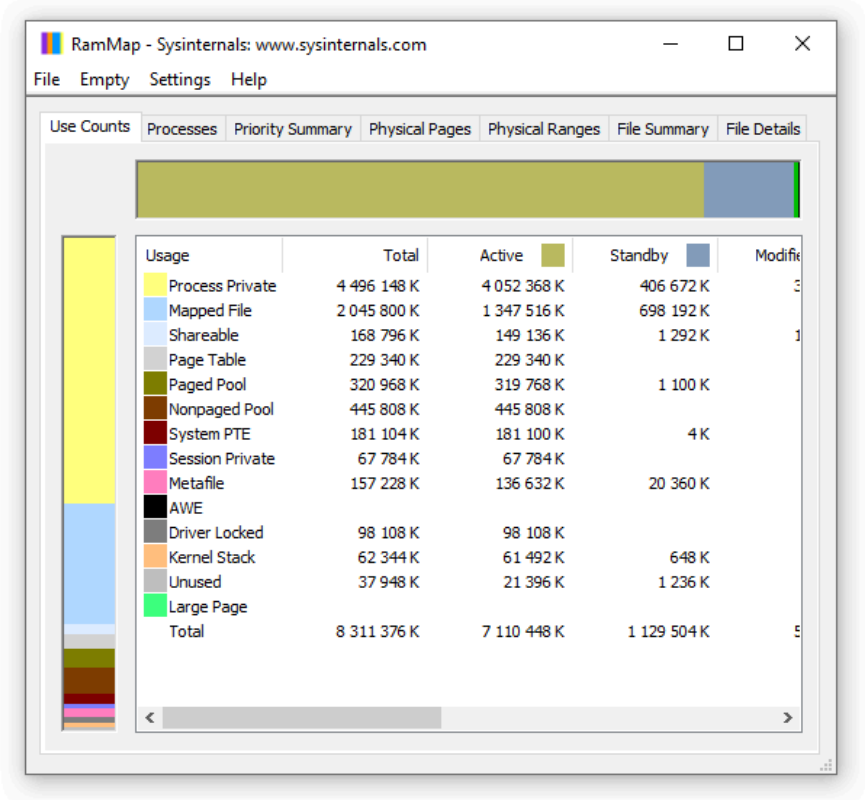
5. Provedeme instalaci serverového agenta, který je dostupný na adrese https://github.com/undera/perfmon-agent/blob/master/README.md. Jedná se o soubor ServerAgent-2.2.3.zip (https://github.com/undera/perfmon-agent/releases/download/2.2.3/ServerAgent-2.2.3.zip)

6. Spustíme serverového agenta dávkou startAgent.bat



Zdroj: Screenshot serverAgent.bat

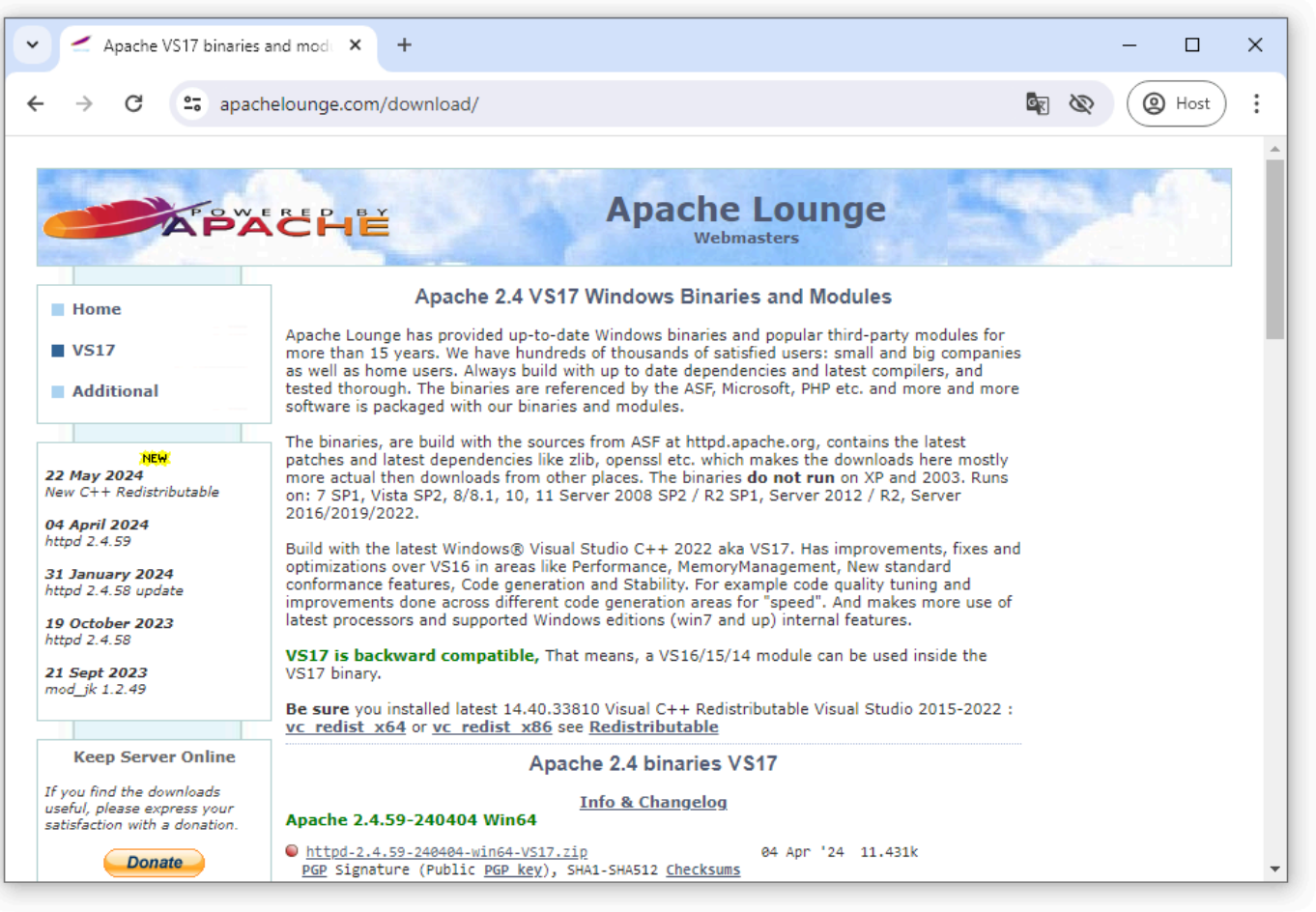
7. Pro práci s pamětí v OS Windows si stáhneme nástroj RAMMap, což je pokročilý nástroj pro analýzu využití fyzické paměti pro Windows Vista a vyšší. Nástroj je k dispozici na stránce https://learn.microsoft.com/en-us/sysinternals/downloads/rammap.



Zdroj: Screenshot aplikace RAMMap

8. Vymazaní cache paměti můžeme provést v menu nástroje RAMMap položkami **Empty → Empty Standby list**

9. Spustíme si vlastní webový server Apache, který si stáhneme z https://www.apachelounge.com/download/.¨Server si rozbalíme do námi zvoleného adresáře a spustíme pomocí dávky {apache_root}/bin/http.exe. Do adresáře {apache_root}/htdosc si můžeme nasadit libovolnou aplikaci.

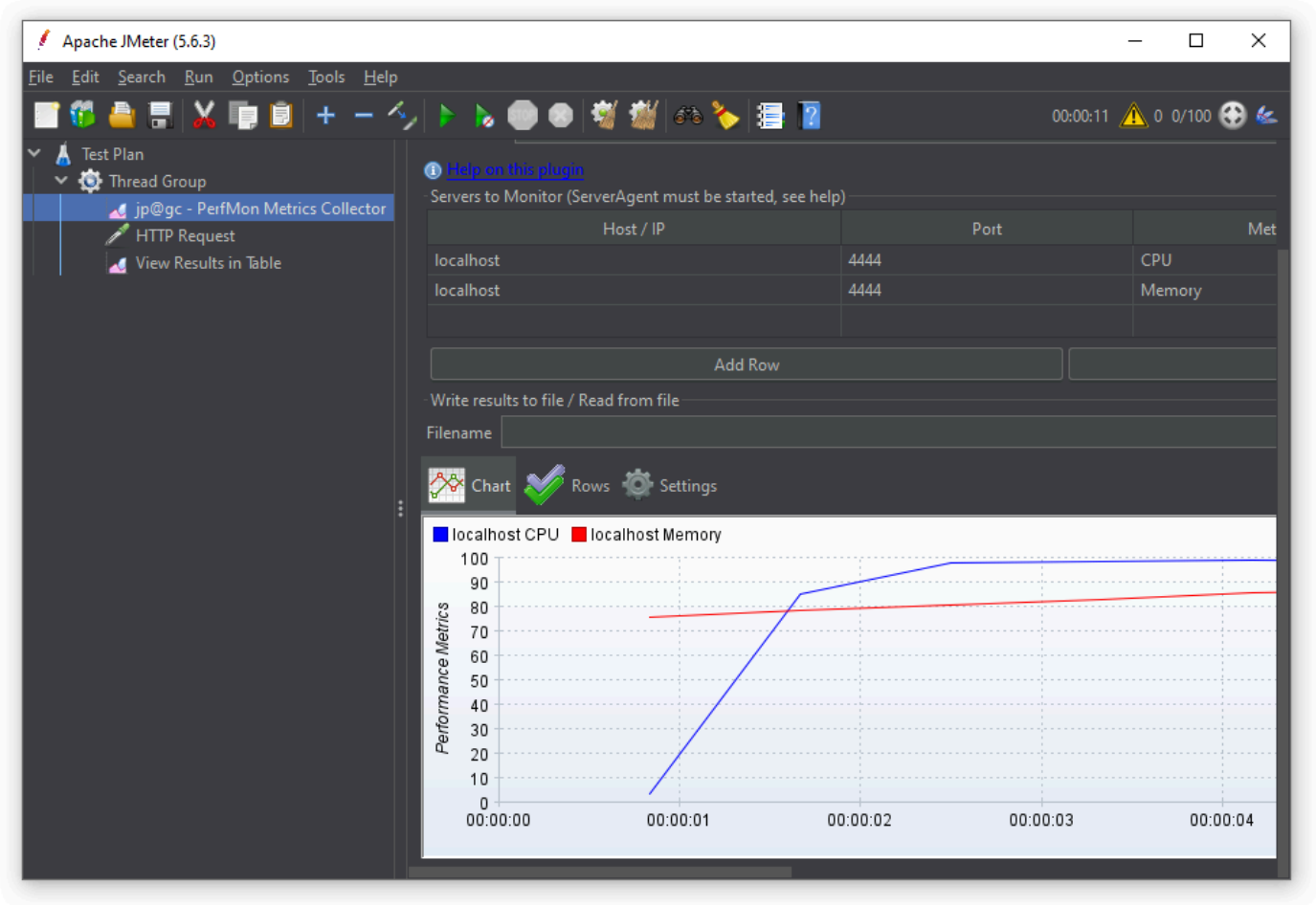


Zdroj: Screenshot stránky https://www.apachelounge.com/

10. Vrátíme se do aplikace JMeter a změníme si parametry parametry testu (Thread Group) na:

- Number of Threads (users): 100
- Ramp-up period (seconds): 10
- Loop Count: 1000

11. Spustíme test a následně si zobrazíme výsledků měření hodnot CPU a paměti v PerfMon Metrics Collector.



Zdroj: Screenshot aplikace JMeter

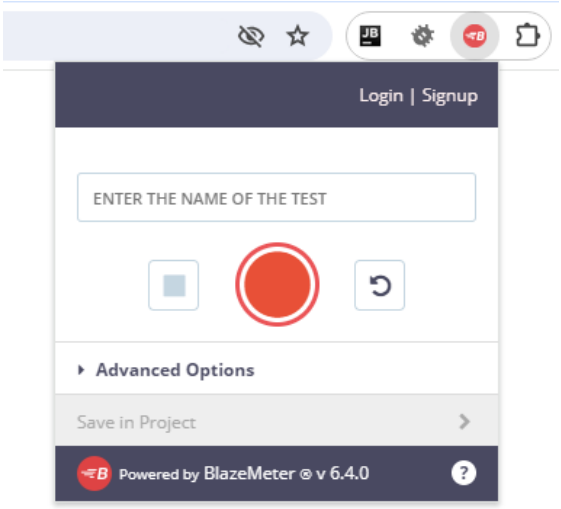
Poznámka: v případě, že vám bude serverAgent padat z důvodu chyby v sigar-amd64-winnt.dll, nahraďte {serverAgent_root}/bin/sigar-amd64-winnt.dll novým z https://github.com/cnstar9988/sigar/raw/master/sigar-amd64-winnt.dll.

12. Následně upravíme definujeme testovací scénář pro námi nasazenou aplikaci a dle tohoto scénáře vytvoříme testovací skript. Při tvorbu testovacího skriptu nám mohou pomoci další aplikace, jako je například níže uvedený BlazeMeter.

BlazeMeter Script Recorder

BlazeMeter Chrome Extension

1. Na stránce https://www.blazemeter.com/ si vytvoříme uživatelský účet.
2. Na stránce https://chromewebstore.google.com/detail/blazemeter-the-continuous/mbopgmdnpcbohhpnfglgohlbhfongabi si přidáme do webového prohlížeče Google Chrome rozšíření BlazeMeter Chrome Extension.



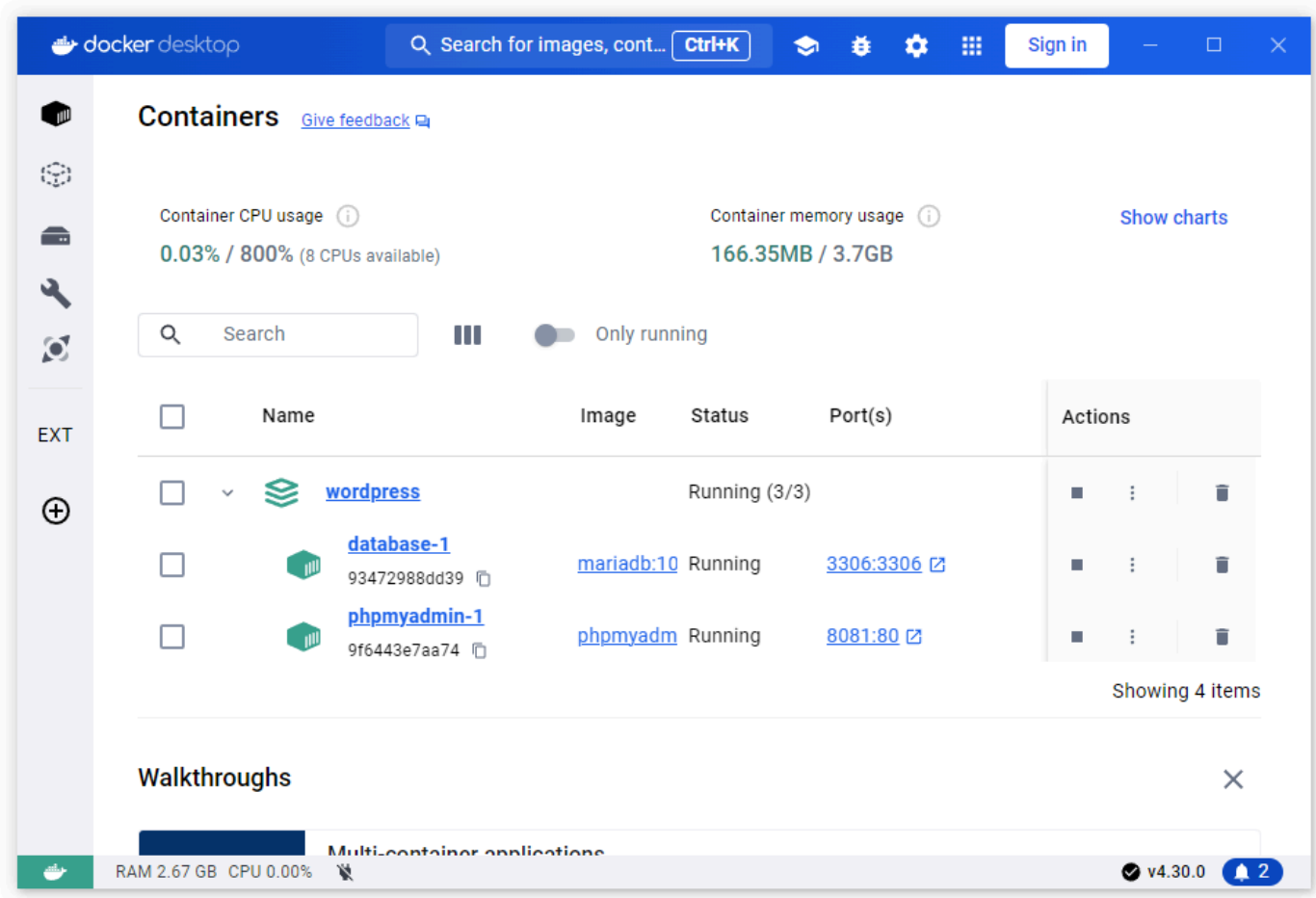
Zdroj: Screenshot BlazeMeter Chrome Extension

Instalace WordPressu pro testování

1. Vytvoříme nový adresář {wordpress_root}, ve kterém vytvoříme nový soubor docker-compose.yml s následujícím kódem:

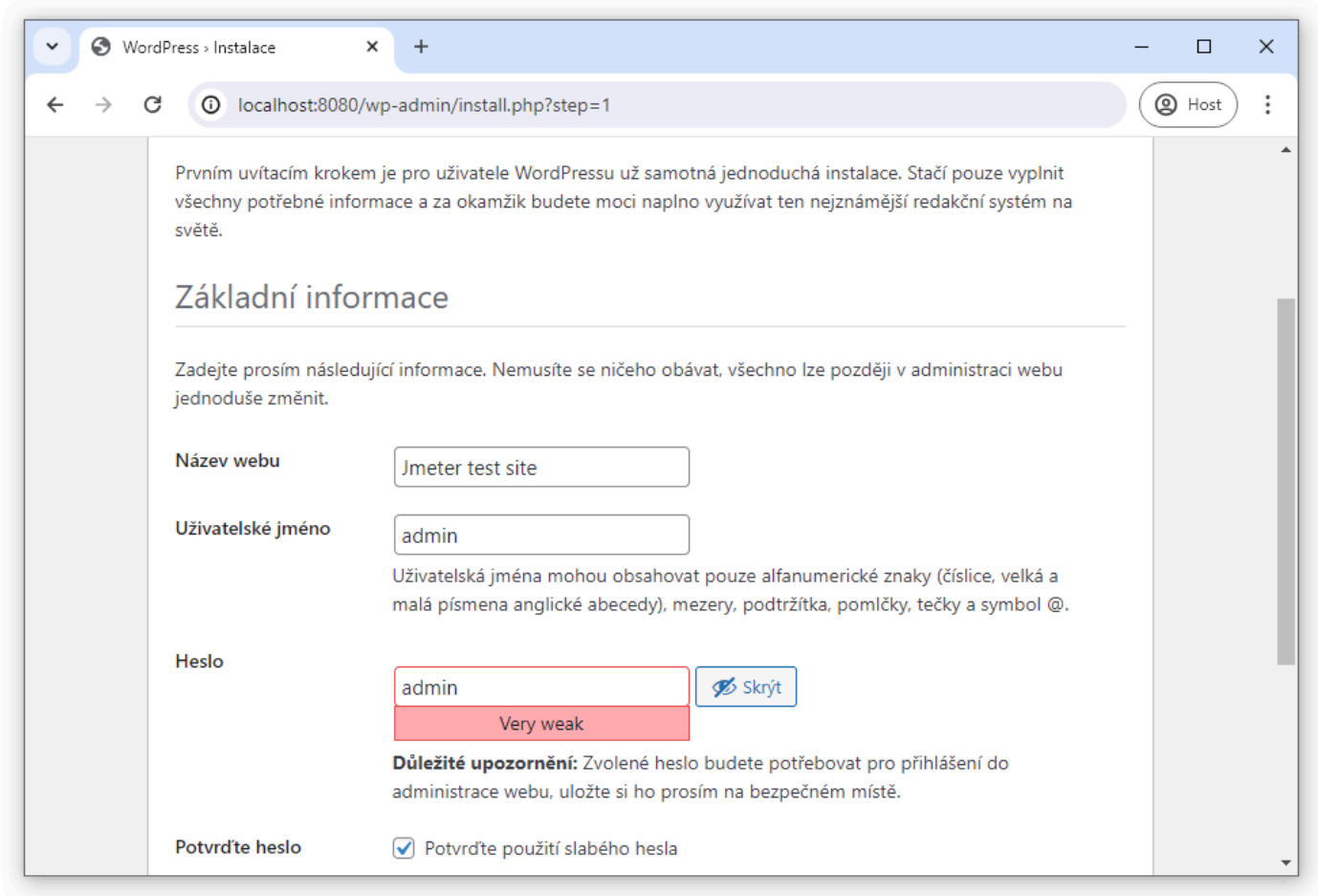
```
services:
  database:
    image: mariadb:10.6.4-focal
    restart: unless-stopped
    ports:
      - 3306:3306
    env_file: .env
    environment:
      MYSQL_DATABASE: '${MYSQL_DATABASE}'
      MYSQL_USER: '${MYSQL_USER}'
      MYSQL_PASSWORD: '${MYSQL_PASSWORD}'
      MYSQL_ROOT_PASSWORD: '${MYSQL_ROOT_PASSWORD}'
    volumes:
      - db-data:/var/lib/mysql
    networks:
      - wordpress-network
    deploy:
      resources:
        limits:
          memory: 2048m

  phpmyadmin:
    depends_on:
      - database
    image: phpmyadmin/phpmyadmin
    restart: unless-stopped
    ports:
      - 8081:80
    env_file: .env
    environment:
      PMA_HOST: database
      MYSQL_ROOT_PASSWORD: '${MYSQL_ROOT_PASSWORD}'
    networks:
      - wordpress-network

  wordpress:
    depends_on:
      - database
    image: wordpress:6.2.2-apache
    restart: unless-stopped
    ports:
      - 8080:80
    env_file: .env
    environment:
      WORDPRESS_DB_HOST: database:3306
      WORDPRESS_DB_NAME: '${MYSQL_DATABASE}'
      WORDPRESS_DB_USER: '${MYSQL_USER}'
      WORDPRESS_DB_PASSWORD: '${MYSQL_PASSWORD}'
    volumes:
      - ./:/var/www/html
    networks:
      - wordpress-network

volumes:
  db-data:

networks:
  wordpress-network:
    driver: bridge
```

2. Příkazem **docker-compose up** spustíme definované kontejnery.



Zdroj: Screenshot Docker Desktop

3. Spustíme adresu localhost:8080 a provedeme nastavení aplikací Wordpress.



Zdroj: Screenshot konfigurace Wordpress

4. V administraci Wordpressu vybereme šablonu Twenty Twenty-One a použijeme.

Testování Wordpresu

1. Do pluginu BlazeMeter se přihlásíme a zadáme název testu

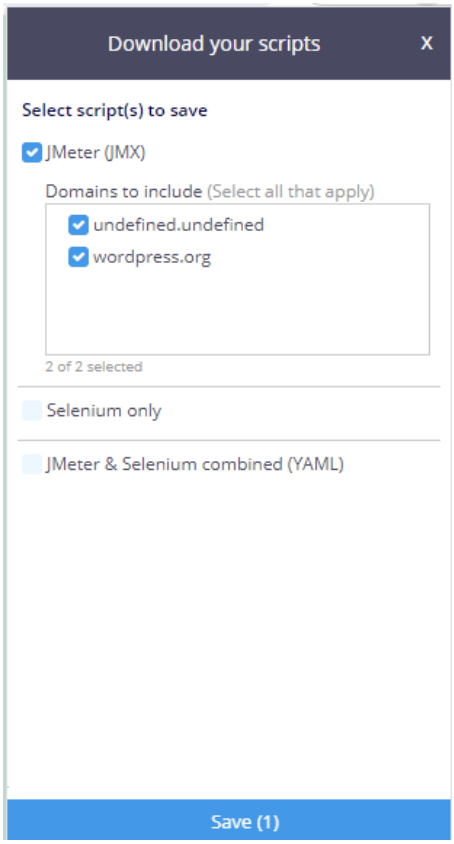
2. Spustit nahrávání tlačítkem **Start recording**

3. Ve výchozím nastavení je každý testovací krok, který se provede, zaznamenán do „Test Case“, což je monolitický skript s posloupností akcí. Na úrovni JMX to jednoduše vytvoří skupinu vláken skládající se z těchto akcí.

4. Proveďte akce dle testovacího scénáře.

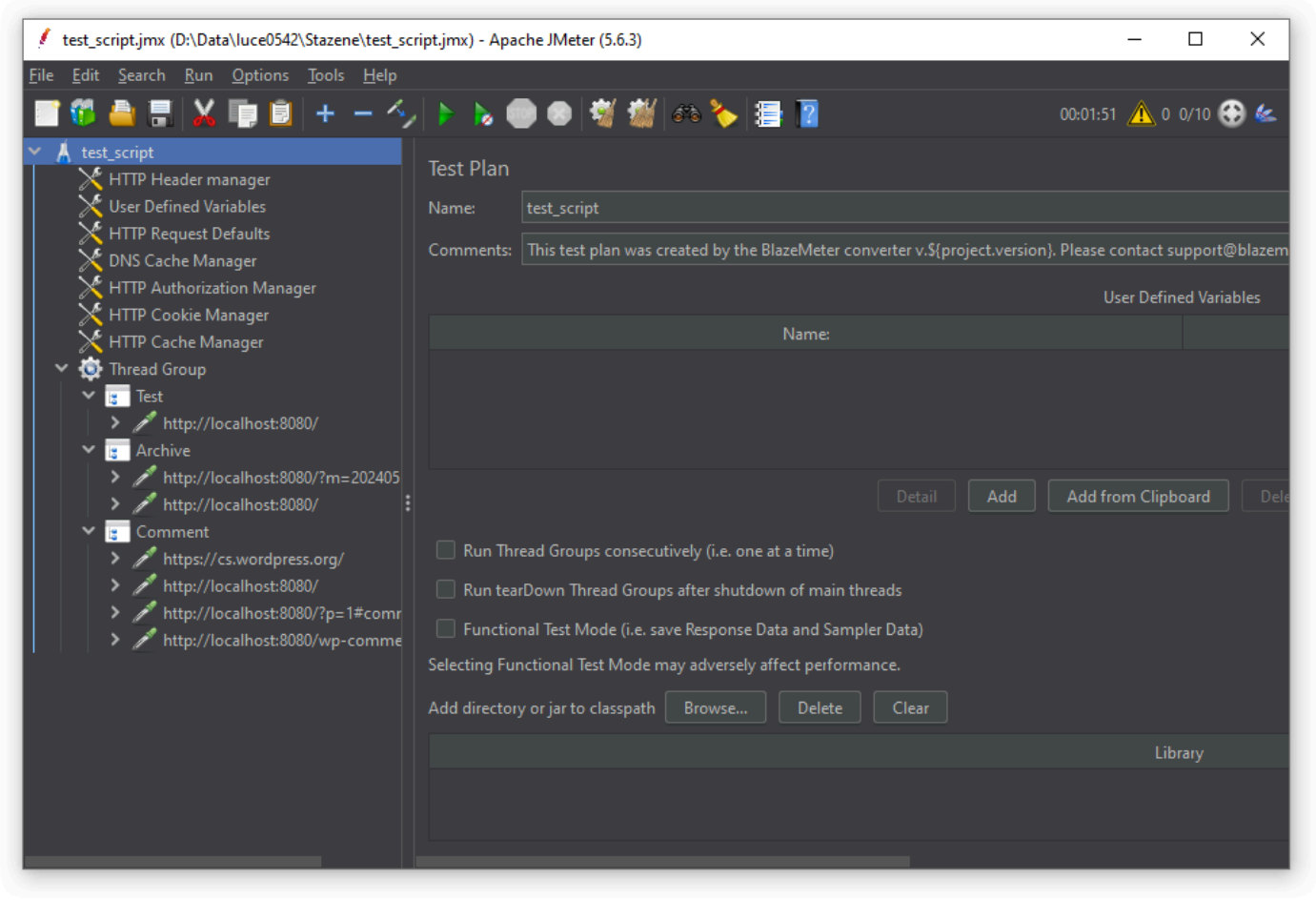
5. Ukončete provádění testu tlačítkem **Stop recording.**

6. Uložte si vytvořený skript JMX a následně ho načtěte v aplikaci JMeter.



Zdroj: Screenshot BlazeMeter Chrome Extension

7. Nyní můžeme v aplikaci JMeter načíst uložený soubor se skriptem testu a testovací skript opakovaně spouštět (v konzoli Dockeru lze sledovat jednotlivé http requesty na server s Worldpressem).



Zdroj: Screenshot aplikace JMeter

JMeter Selenium WebDriver

1. Instalace pluginu Selenium/WebDriver (homepage GitHub - undera/jmeter-plugins-webdriver: WebDriver Components for Apache JMeter)

2. Vytvoříme nový testovací plán a přidáme Add - Threads (Users) - Thread Group

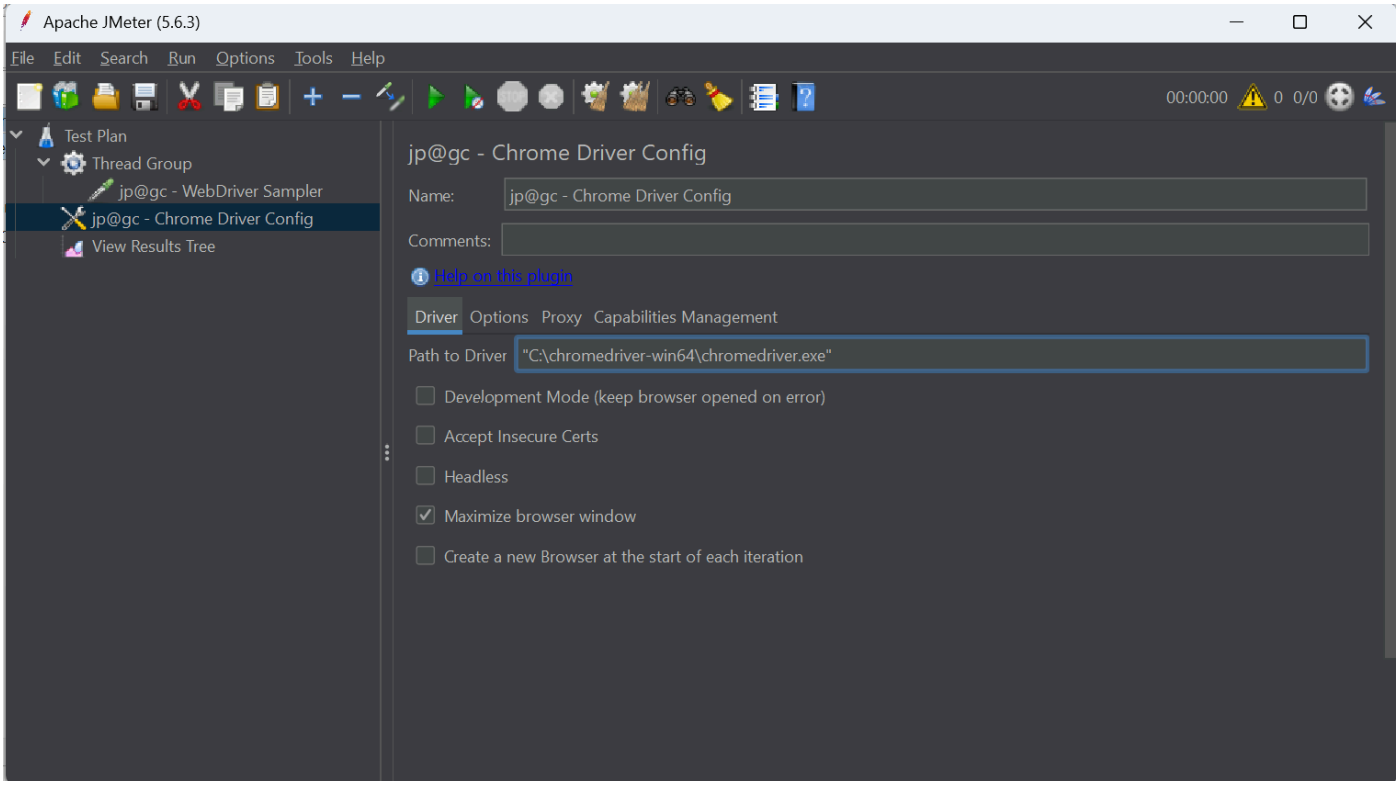
3. Přidáme Add - Config Element - jp@gc - Chrome Driver Config

4. Přidáme sampler ThreadGrou - Add - Samper - jp@gc Web Driver Sampler

5. Add - Listener - View Results Tree

6. Stáhneme si chromdriver.exe (https://developer.chrome.com/docs/chromedriver/downloads) (https://storage.googleapis.com/chrome-for-testing-public/125.0.6422.141/win64/chromedriver-win64.zip)

7. Na kartě jp@gc - Chrome Driver Config přidáme cestu k driveru.



7. Přidáme níže uvedený skript na kartě jp@gc Web Driver Sampler

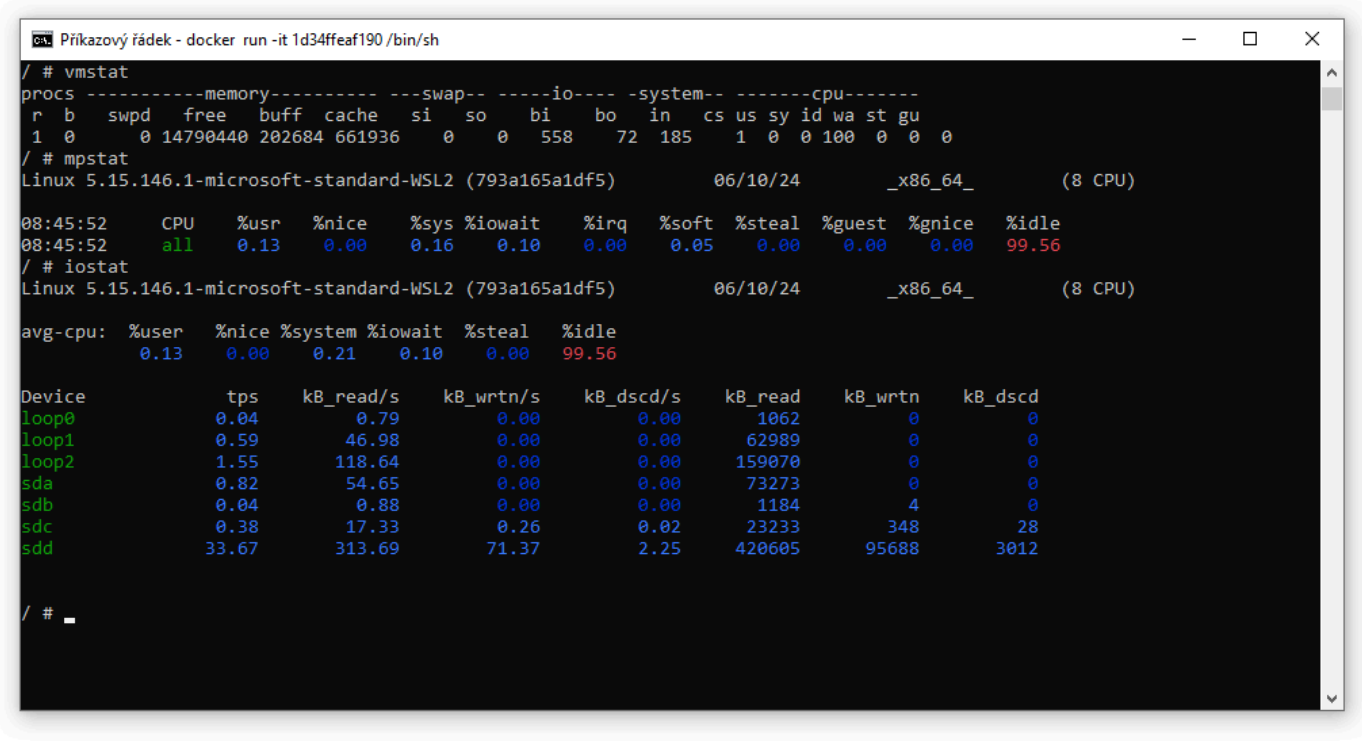
```
WDS.sampleResult.sampleStart()
WDS.browser.get('https://www.upce.cz')
WDS.sampleResult.sampleEnd()
```

8. Spustíme test a analyzujeme výsledek testu na kartě View Results Tree.

Monitoring systémových zdrojů v průběhu testu

Pro online monitoring systémových zdrojů můžeme na Linuxových OS použít nástroje:

- vmstat - shromažďuje statistiky o využití paměti, procesech, stránkování, přerušení a blokování
- mpstat - shromažďuje statistiky o využití CPU a výkonu
- iostat - shromažďuje statistiky o vstupech a výstupech úložiště

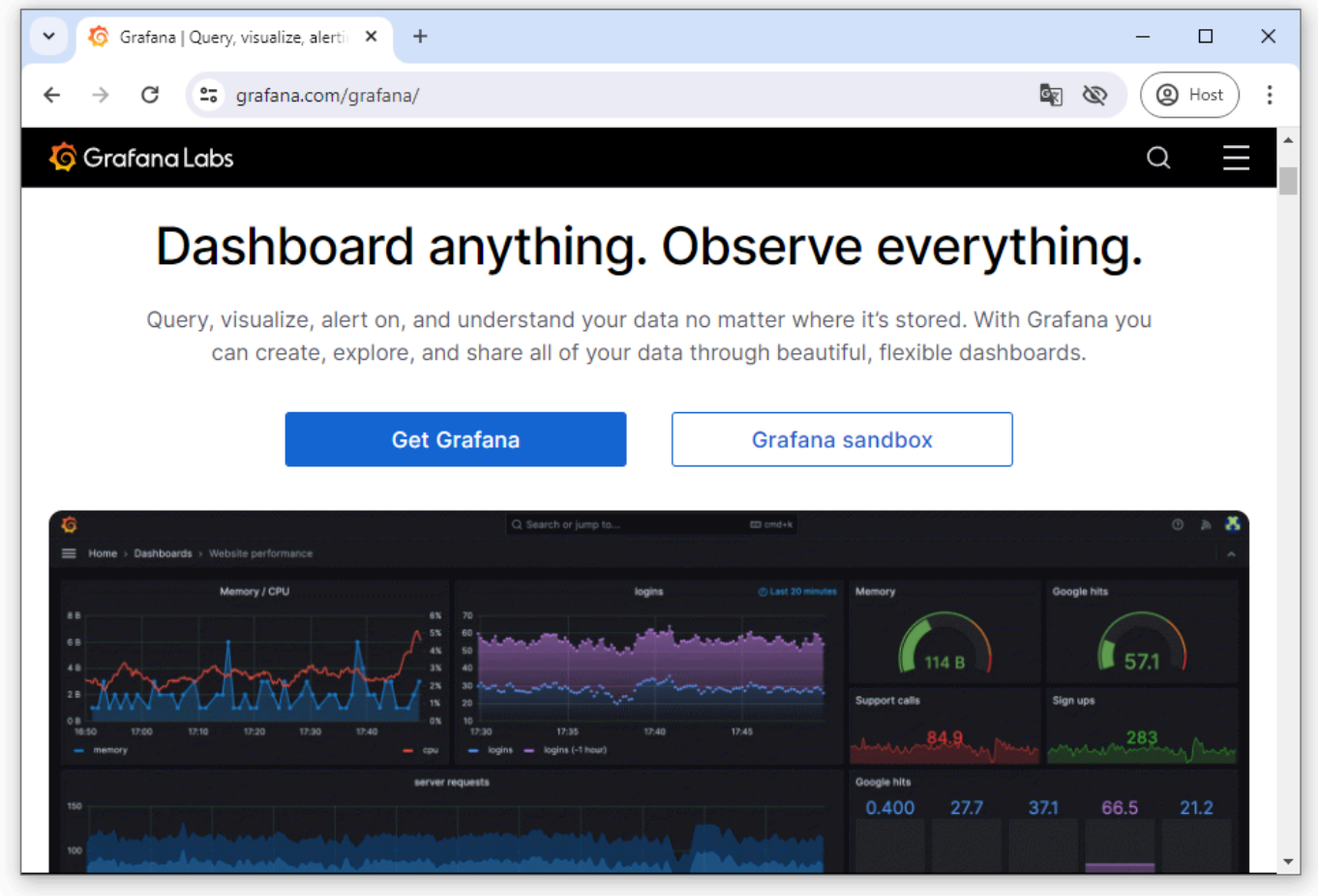


Zdroj: Screenshot vmstat, mpstat, iostat

Monitoring serveru pomocí nástrojů Prometheus a Grafana

Grafana

Grafana je nástroj pro analýzu a monitorování dat v reálném čase. Umožňuje vám dotazovat se, vizualizovat, upozorňovat a lépe porozumět metrikám, a to bez ohledu na to, kde jsou uloženy.



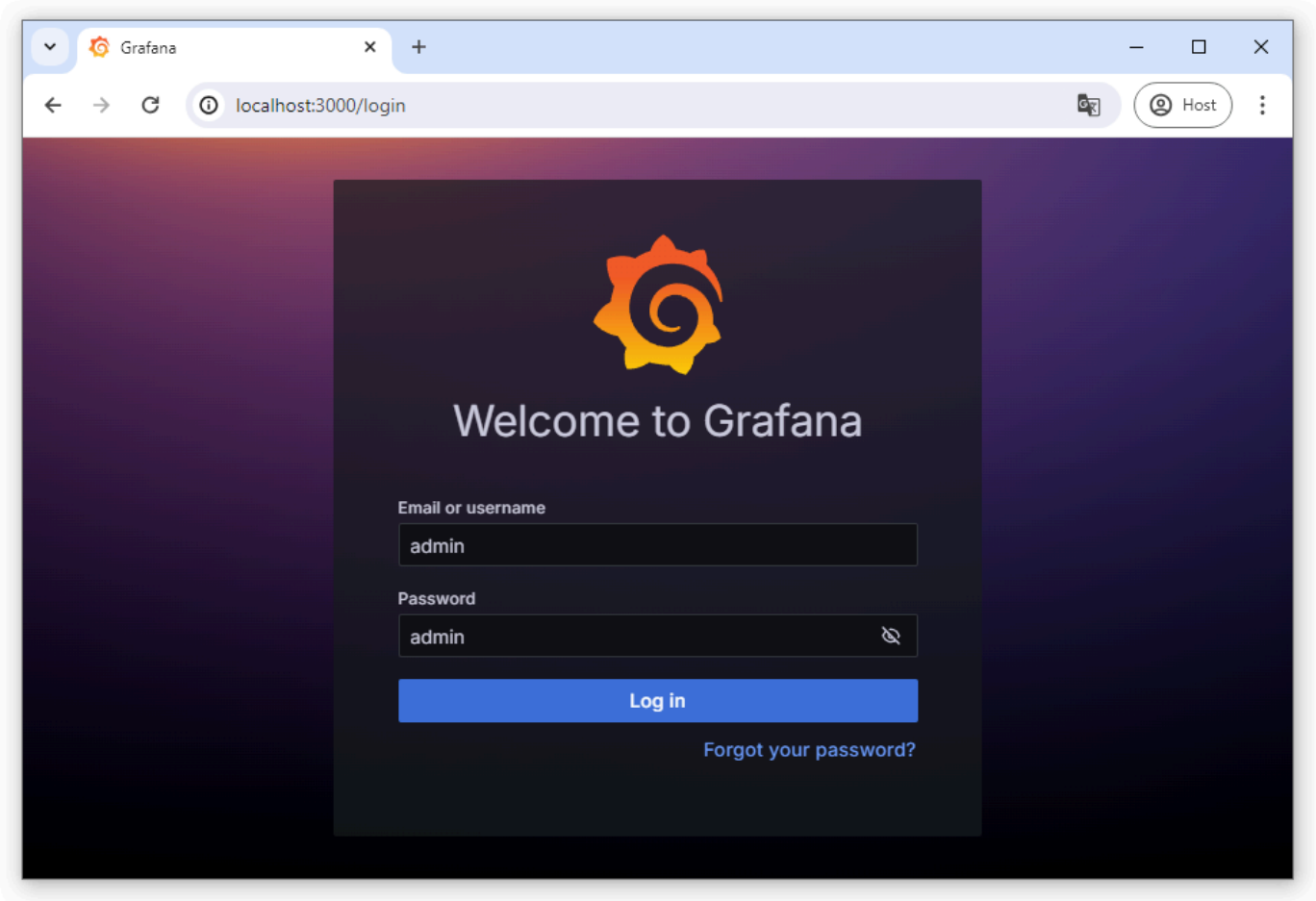
Zdroj: Screenshot https://grafana.com/grafana

1. Grafanu si spustíme v kontejneru Docker: docker run -d -p 3000:3000 --name grafana grafana/grafana:latest

2. Ověříme si, že kontejner je spuštěn: docker ps

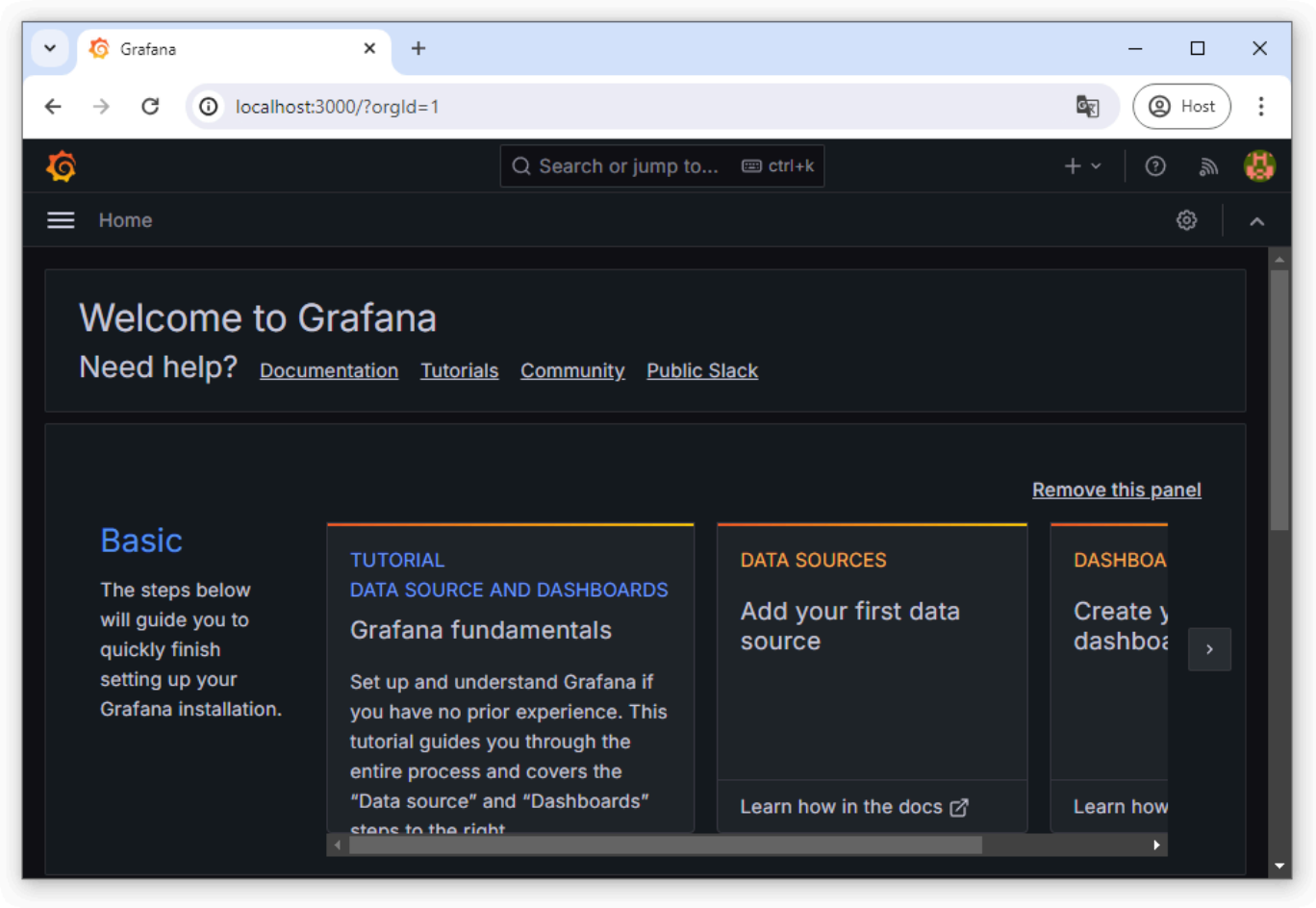
3. Ve webovém prohlížeči otevřeme lokální instanci Grafany: localhost:3000

4. Zadáme výchozí přihlašovací údaje: username: admin, password: admin.



Zdroj: Screenshot https://localhost:3000

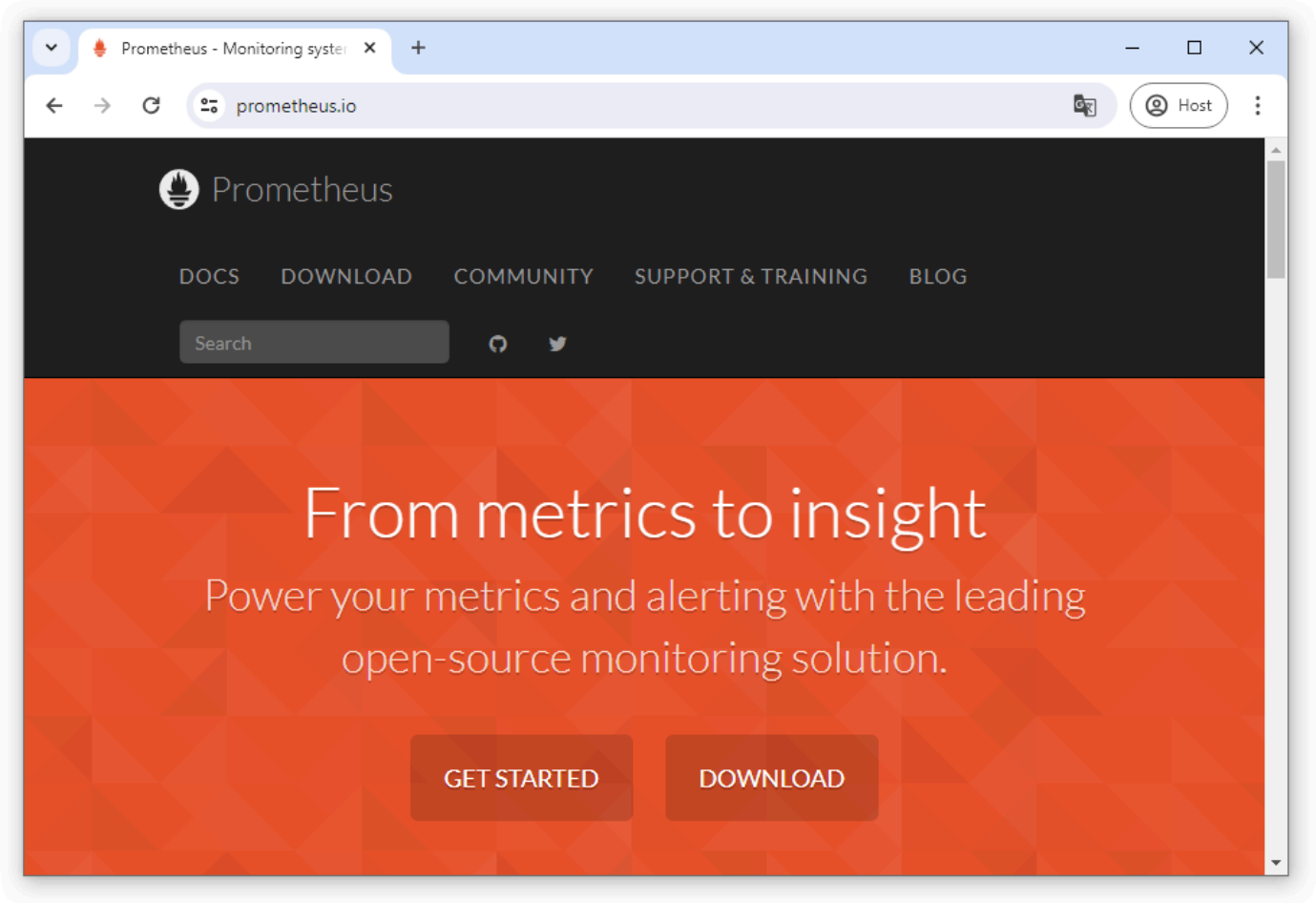
5. Po přihlášení se můžeme seznámit s Grafanou



Zdroj: Screenshot https://localhost:3000

Prometheus

Prometheus je monitoringový nástroj a metrický server. Umožňuje sbírat metrická data z aplikací a většiny komponent provozované infrastruktury a nad těmito daty lze v Grafaně vytvářet grafy.



Zdroj: Screenshot https://prometheus.io

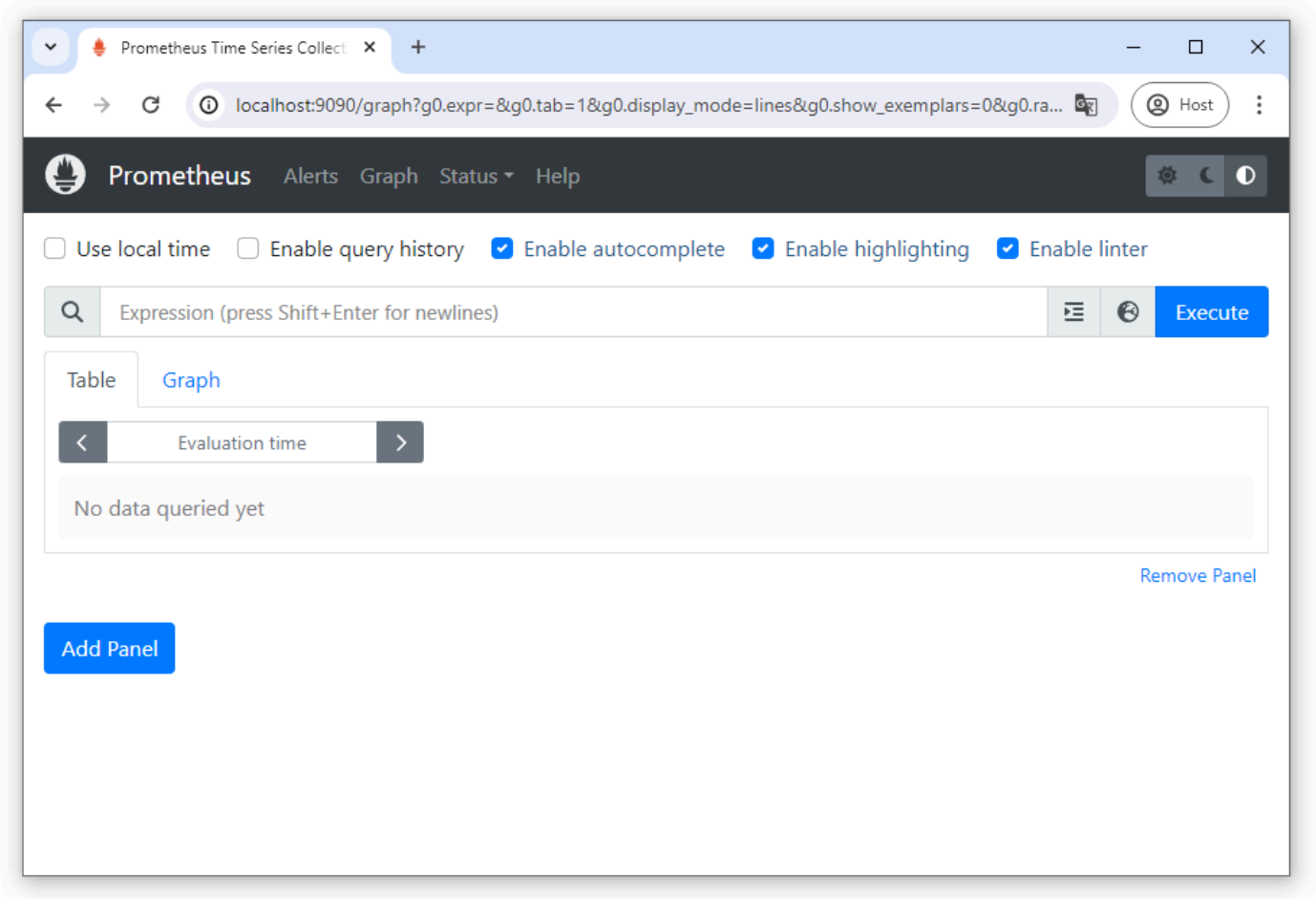
1. Vytvoříme si konfigurační soubor prometheus.yml, do kterého vložíme následující zdrojový kód (viz https://prometheus.io/docs/prometheus/latest/getting_started/):

```
global:
  scrape_interval: 15s
scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 5s
    static_configs:
    - targets: [ 'localhost:9090' ]

  - job_name: 'win11'
    scrape_interval: 5s
    static_configs:
    - targets: [ 'host.docker.internal:9182' ]
```

2. Prometheus si spustíme v kontejneru Docker: docker run -d -p 9090:9090 --name prometheus -v "D:/docker/web_performance_monitoring/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml" prom/prometheus:latest

3. Ověříme si, že kontejner je spuštěn: docker ps

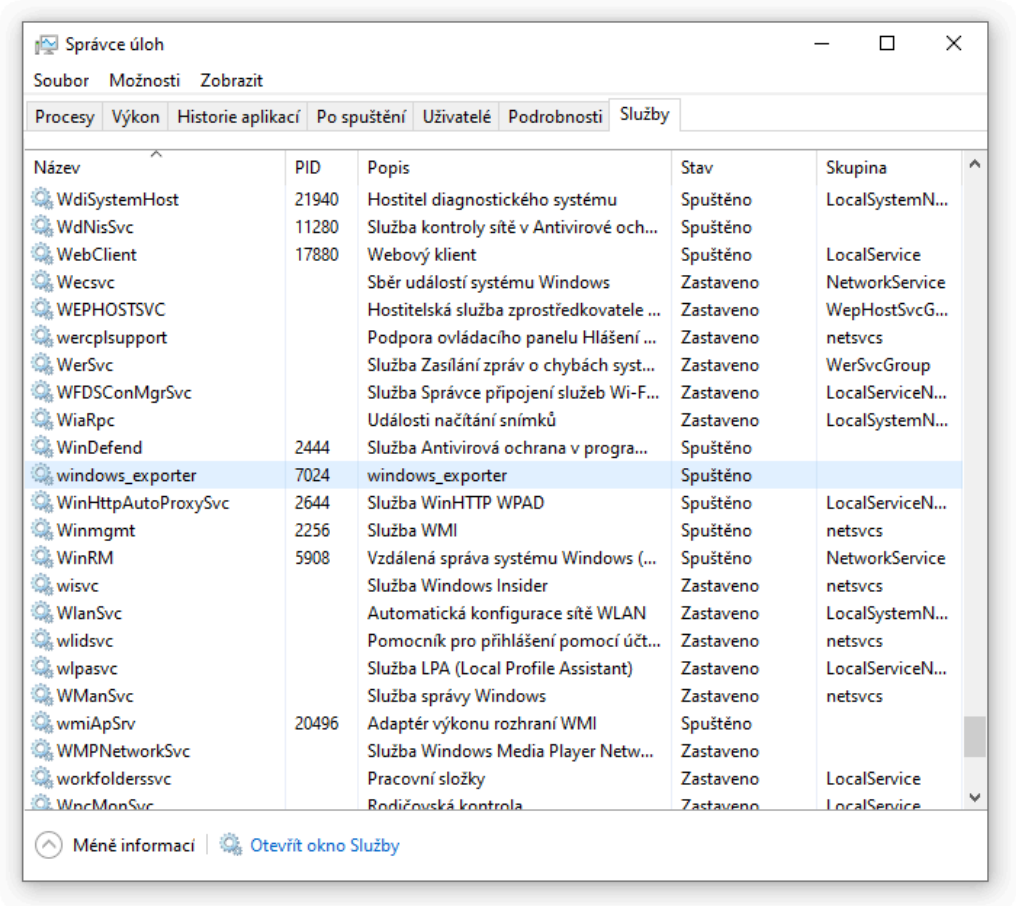
4. Ve webovém prohlížeči otevřeme lokální instanci Promethea: localhost:9090



Zdroj: Screenshot https://localhost:9090

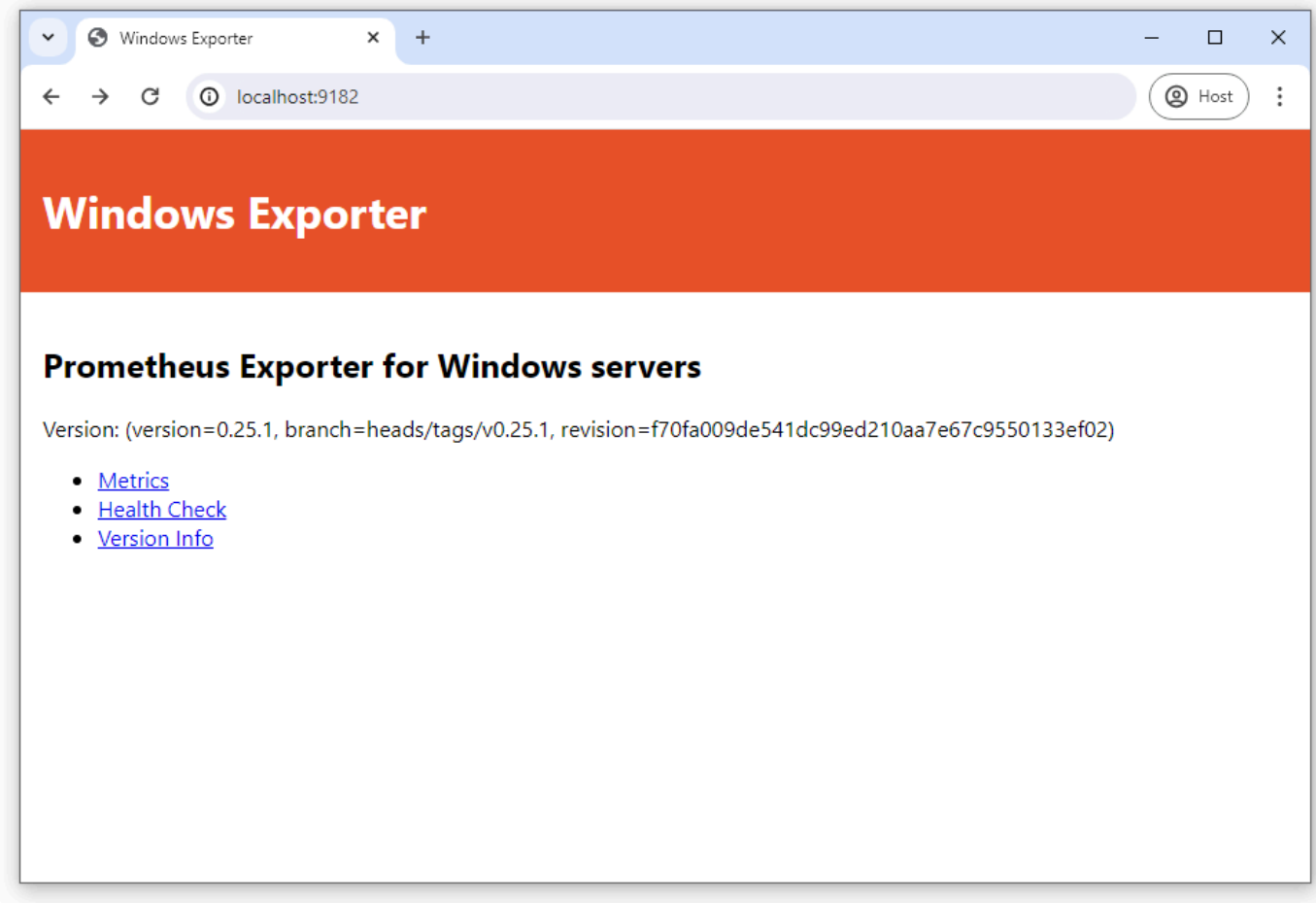
5. Na stránce http://localhost:9090/metrics si můžeme prohlédnou metriky.

6. Na stránce https://github.com/prometheus-community/windows_exporter si stáhneme windows_exporter (poslední release: https://github.com/prometheus-community/windows_exporter/releases/download/v0.25.1/windows_exporter-0.25.1-amd64.msi) Windows exporter je prostředek, který shromažďuje metriky jako je využití CPU, paměti, disku a podobně. Po instalaci bychom měli najít ve Správci úloh běžící službu "windows_exporter".



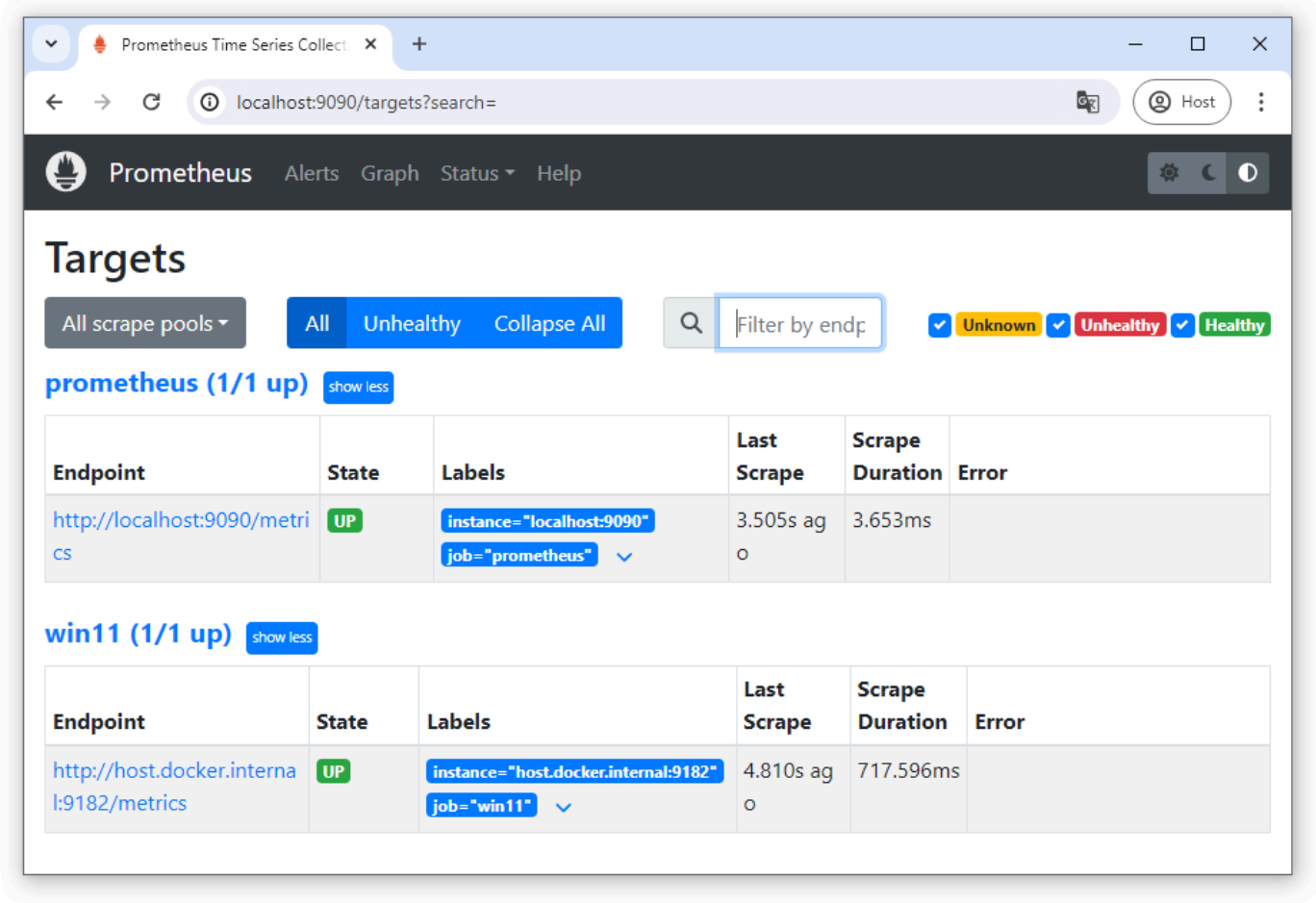
Zdroj: Screenshot Správce služeb

7. Na adrese localhost:9182 si můžeme ověřit běžící Windows exporter.



Zdroj: Screenshot localhost: 9182

8. V Prometheu si můžeme ověřit, že oba zdroje dat běží.

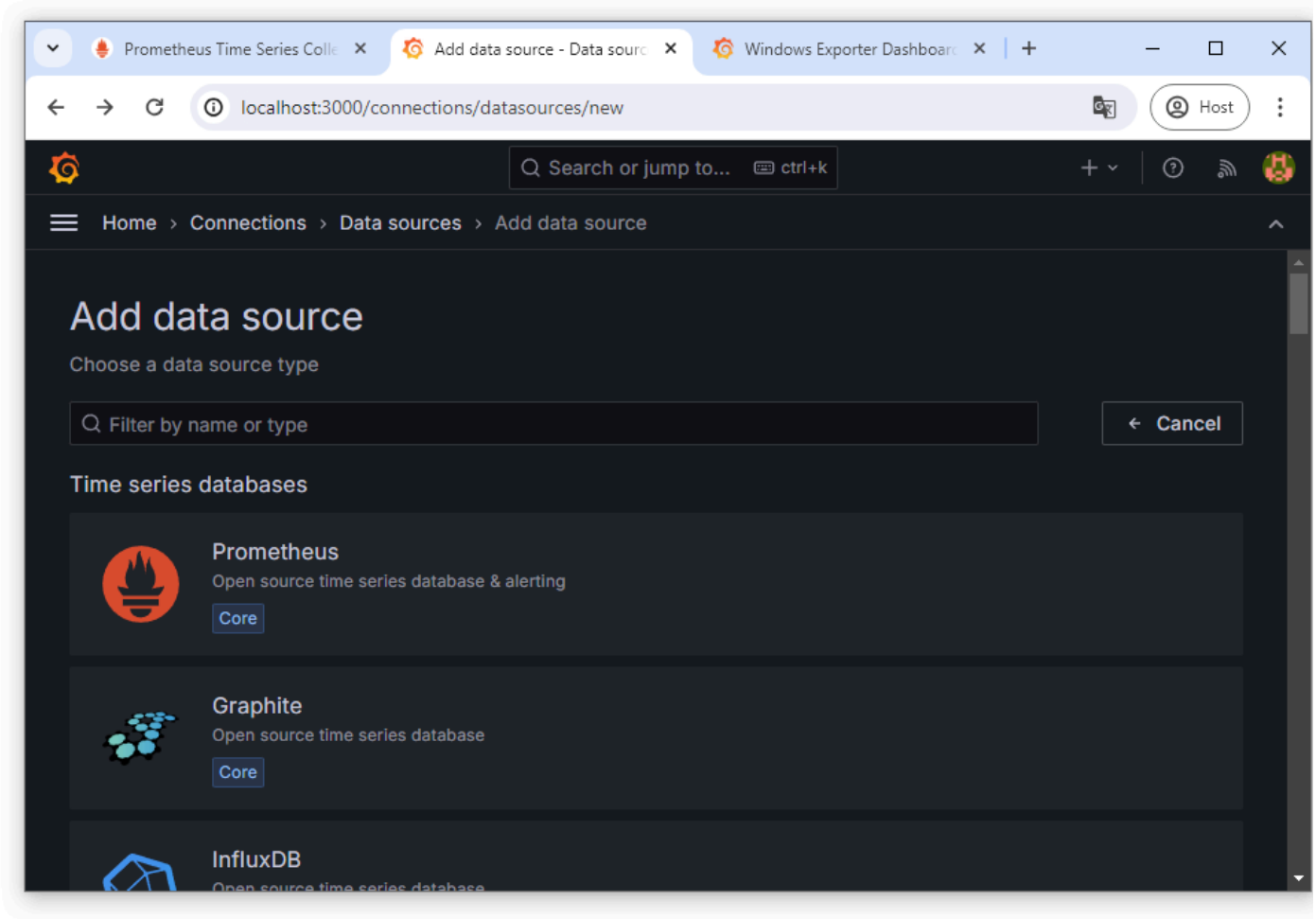


Zdroj: Screenshot Prometheus Targets

Pokud používáte OS linux, tak místo Windows exporteru použíte Node exporter https://hub.docker.com/r/prom/node-exporter

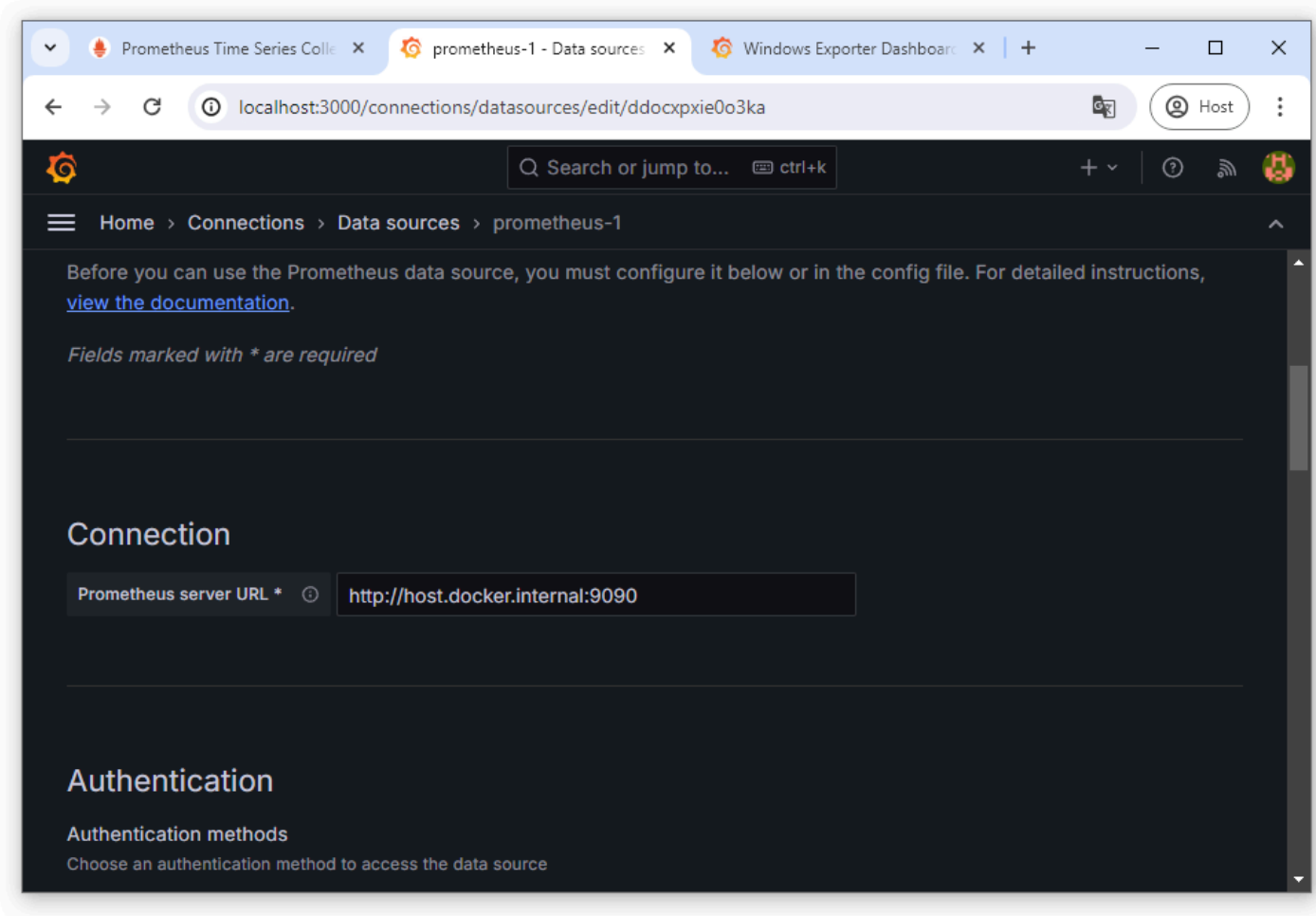
Zobrazení metrik z Promethea v Grafaně

1. Nejdříve si přidáme zdroj dat ze serveru Prometheus. V menuvybereme položku Connections -> Data sources a zvolíme tlačítko "Add data source". Na následující stránce vybereme položku "Prometheus"



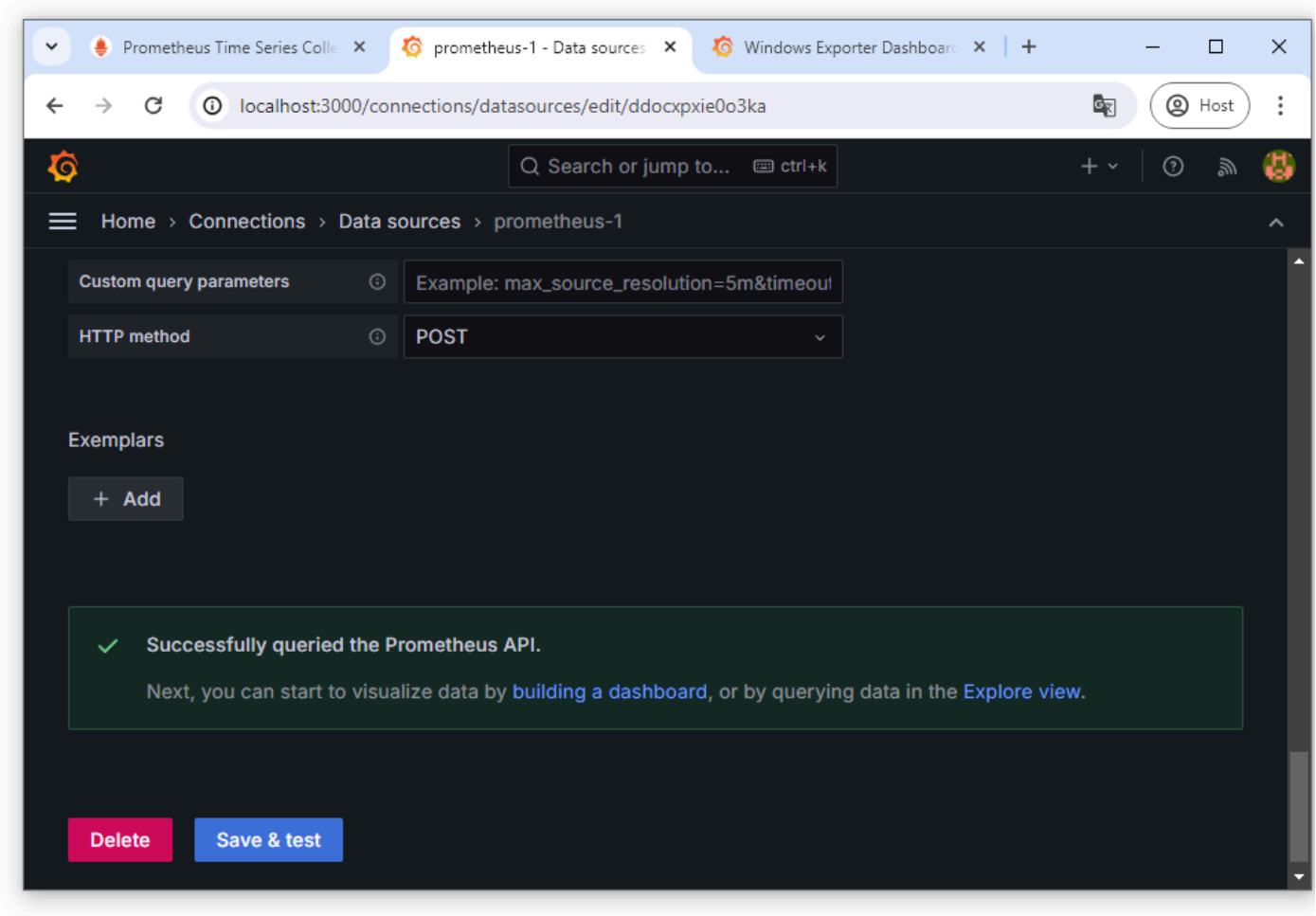
Zdroj: Screenshot Grafana Add data source

2. Do vstupního pole "Prometheus server URL" vložíme adresu serveru: http://host.docker.internal:9090



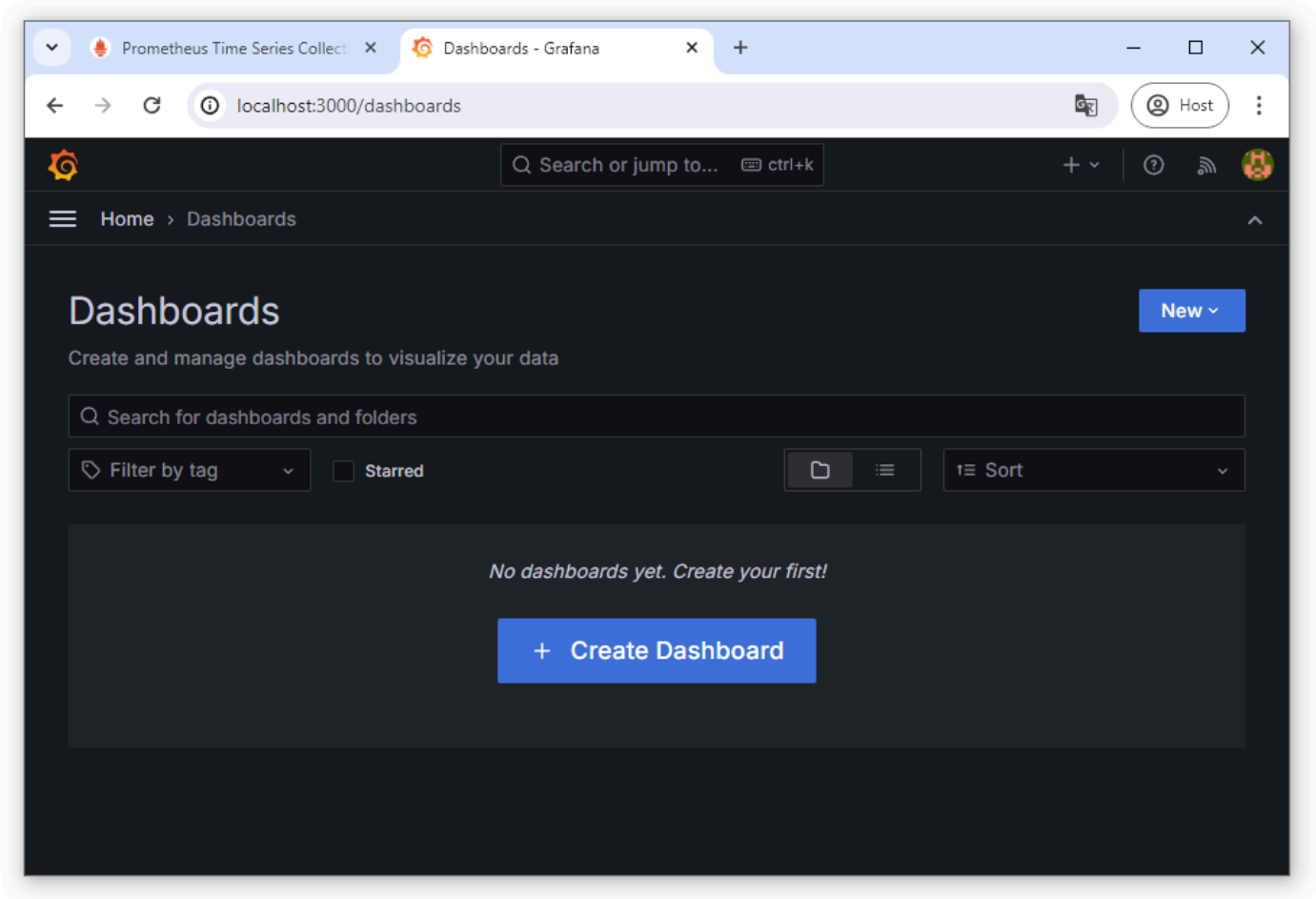
Zdroj: Screenshot Grafana Add data source

3. Následně zvolíme tlačítko "Save & test". Po úspěšném přidání serveru Prometheus bychom měli obdržet zprávu "Successfully queried the Prometheus API.".



Zdroj: Screenshot Grafana Add data source

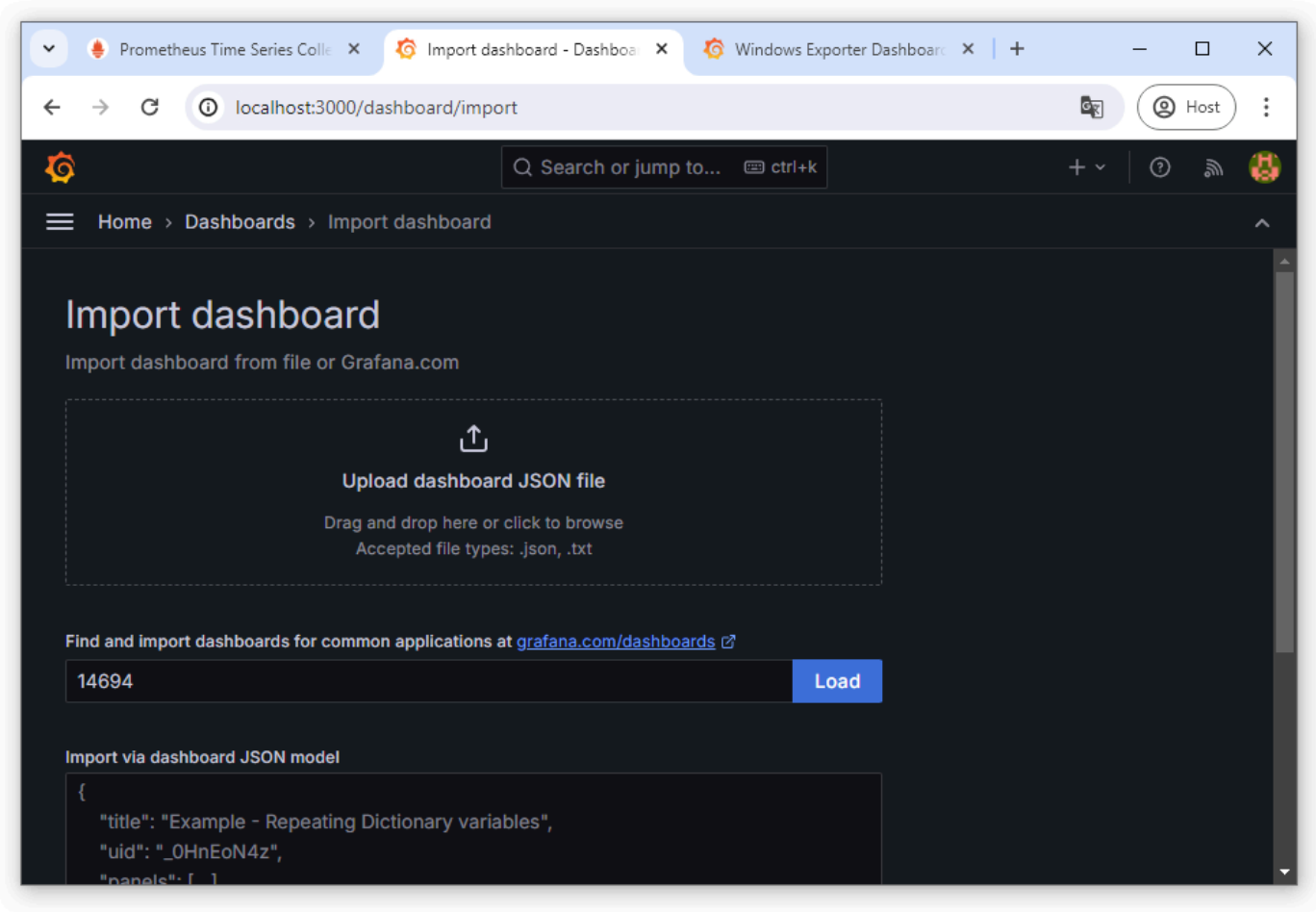
4. V Grafaně si v Menu vybereme položku Dashboards a následně přidáme dashboard pomocí tlačítka "Create Dashboard"



Zdroj: Screenshot Grafana Dashboard

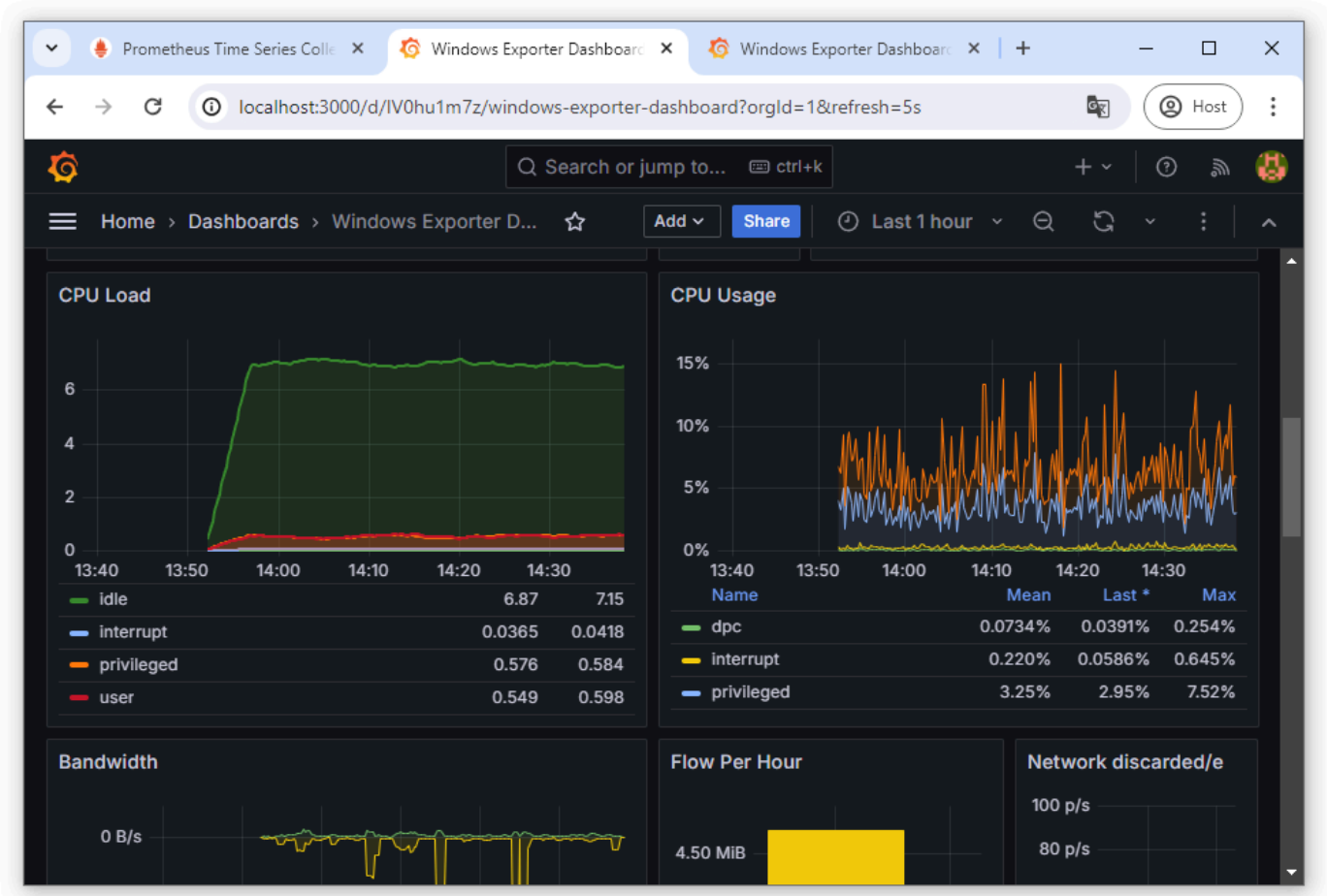
5. Dále importujeme již vytvořený dashboard ze šablony pomocí tlačítka "Import dashboard".

6. Na adrese https://grafana.com/grafana/dashboards/ si můžeme vyhledat vhodný dashboard pro naše použití. V našem příklad použijeme například šablonu "Windows Exporter Dashboard" (https://grafana.com/grafana/dashboards/14694-windows-exporter-dashboard/), jehož id je "14694", které vložíme do řádku "Find and import dashboards for common applications at grafana.com/dashboards"



Zdroj: Screenshot Grafana Import dashboard

7. Pro zvolenou šablonu si vybereme zdroj dat, který jsme si přidali do Grafany v předcházejících krocích.



Zdroj: Screenshot Grafana Windows Exporter Dashboard

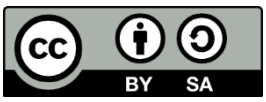
◄ Teoretická východiska laboratorní úlohy

Přejít na...

Otestování znalostí ►

UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Vytvořeno v rámci projektu **Digitalizace studijních Agend, Nové TechnologiE, systémy a přístupy k výuce na UPCE**, reg. č. NPO_UPCE_MSMT-16591/2022.



Financováno Evropskou unií
NextGenerationEU

Národní plán obnovy

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY