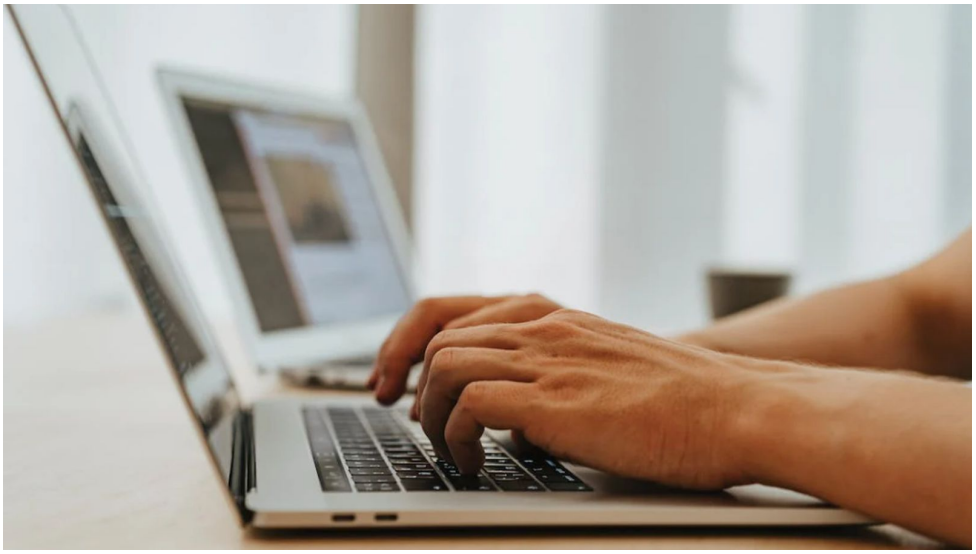
SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

CTF systém v Kubernetes



Autor: Jan Stránský

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2024/25

Poděkování

Rád bych poděkoval pánům učitelům Ing. Petru Grussmannovi a Mgr. Marku Lučnému za jejich pomoc s projektem, jelikož mi poskytovali cenné rady a připomínky.

Prohlášení

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým a prezentačním účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 1. 1. 2025

. . . . . .

Podpis autora

Abstrakt

Výsledkem tohoto projektu je funkční systém pro spouštění a vytváření úloh CTF typu v systému Kubernetes běžícím na školní síti s dostatečnou mírou zabezpečení. Aplikace zahrnuje registraci a přihlašování uživatelů, zapínání nových úloh a následně jejich vypínání. Hlavní částí tohoto projektu je komunikace se systémem Kubernetes, který se využívá ve vysoce škálovaných produkčních prostředích. Uživatel s aplikací může komunikovat skrz poskytnuté webové prostředí, ale může komunikovat i přímo s poskytnutou API. Dále si tento projekt klade za cíl umožnit studentům se lépe seznámit s určitými možnostmi v oblasti IT formou hry (CTF) jako to dělají služby jako např. TryHackMe nebo HackTheBox.

Klíčová slova

CTF, Kubernetes, FastAPI, webová aplikace

Abstract

Keywords

Template, LaTeX, High school proffessional activity, ...

Obsah

Úvod

Mým cílem v této práci bylo sestavit škálovatelný software, který by nad prostředím Kubernetes vytvářet a spravovat kontejnery pro soutěž typu CTF (Capture The Flag). Zároveň bylo cílem, aby se tento software dal nasadit i v prostředí s nízkým oprávněním a aby ho šlo škálovat díky architektuře mikroslužeb.

Hlavní motivací bylo pochopení funkce a komunikace v rámci aplikací s formátem typu mikroslužeb místo monolitických aplikací a zlepšení svých dovedností v oblasti prostředí Kubernetes.

Zvláštní zaměření bylo na backendovou část API a na zabezpečení celého systému.

1 Backend mikroslužby

1.1 Úvod

V této kapitole se seznámíme s tím, co mikroslužby jsou a s jednotlivými mikroslužbami použitými v API částí tohoto projektu. Všechny tyto mikroslužby jsou napsány v jazyce Python s použitím

Tyto mikroslužby jsou:

- Router
- Auth
- Lister
- Deployer
- Deleter
- Flag-submitter

1.2 Router

Tato mikroslužba je zodpovědná za směrování požadavků na správné mikroslužby a veškeré požadavky na API putují skrz ni, díky čemuž se dá využít globální modifikace, monitorování a logování požadavků. Kvůli tomuto účelu tato služba nepotřebuje žádné privilegované přístupy do ostatních částí systému. Jednou z částí této mikroslužby je i zajištění přesunu JWT tokenu z cookie do hlavičky požadavku, aby se dala API používat jak z webového frontendu, tak i z jiných aplikací.

Tato mikroslužba zároveň funguje jakožto filtr nevalidních typů požadavků (dále posílá pouze požadavky typu GET, POST, PUT a DELETE, ostatní jsou zahozeny s chybovou hláškou)

1.3 AUTH

Tato mikroslužba je zodpovědná za registraci uživatele a vytvářením jeho záznamu v databázi PostgreSQL.

Tato služba je jediná, která má přístup k privátnímu klíči používaného k podepisování tokenů algoritmem RS256. Dále je také zodpovědná za ověření přihlašovacích údajů uživatele a vytvoření JWT tokenu, který se následně používá pro ověření uživatele v ostatních částech systému. Tato služba má přístup k databázi PostgreSQL.

Tato služba má tři API endpointy:

- POST /register
- POST /login
- GET /health

kde první dva slouží k registraci a přihlášení uživatele a třetí slouží k zjištění stavu služby, primárně kvůli liveness a readiness HTTP checku v Kubernetes při chybě nebo při čekání na databázi.

1.4 LISTER

Účel mikroslužby Lister je umožnění uživatelům získat informace o všech dostupných úlohách a jejich stavech. Dále tato služba umožňuje získat data o právě aktivních úlohách uživatele a získání detailních informací o těchto úlohách.

Tato mikroslužba potřebuje přístup k Redis a PostgreSQL databázím.

Tato služba má čtyři API endpointy:

- GET /
- GET /running
- GET /running/id
- GET /health

kde první endpoint vrací veškeré dostupné úlohy a nepotřebuje žádné přihlášení, zatímco druhý a třetí endpoint vrací informace o právě běžících úkolech uživatele, tudíž vyžadují token, s tím, že třetí vrací i detailní informace o tomto úkolu.

1.5 Deployer

Tato mikroslužba zajišťuje zapínání úkolů uživatele v systému Kubernetes a zapsání informací o této běžící službě do databáze Redis, čímž zpřístupní tato data službě Lister.

Jednotlivé úkoly jsou v Kubernetes spuštěné jakožto pody v namespace daným uživatelem, což je také jeden z důvodů užívání samostatného Kubernetes clusteru (ať už opravdového nebo vcluster) pro tyto studentské stroje - ServiceAccount spojený s tímto projektem musí mít jak práva na vytváření nových podů, tak vytváření nových namespace.

Tato služba vyžaduje přístup k Redis a PostgreSQL databázím a ke Kubernetes API.

Tato služba má dva API endpointy:

- POST /
- GET /health

kde základní endpoint vyžaduje JSON data s `challenge_id` klíčem. Dále tento endpoint potřebuje přístup k tokenu.

1.6 Deleter

Tato mikroslužba umožňuje vypínat (mazat) již vytvořené úkoly uživatele a to jak v Redis databázi, tak jejich instance běžící v systému Kubernetes.

Tato služba vyžaduje přístup k Redis databázi a ke Kubernetes API.

Tato služba má dva API endpointy:

- DELETE /id
- GET /health

kde endpoint `/{id}` vyžaduje id úkolu, který uživatel chce vypnout a JWT token uživatele.

1.7 Flag-submitter

Tato mikroslužba umožňuje odevzdávat řešení jednotlivých úkolů (vlajky).

Tato služba vyžaduje přístup k PostgreSQL databázi.

Tato služba má dva API endpointy:

- POST /flag_id
- GET /health

kde endpoint `/{flag_id}` vyžaduje v těle požadavku string `flag` a token uživatele.

2 FRONTEND

Frontend část tohoto projektu je napsána v Reactu a jakožto nástroje je využíván projekt Vite. Samotná stránka funguje na bázi CSR (Client Side Rendering) a komunikuje s API popsáným v předchozí kapitole. Díky tomuto je tato stránka zároveň kódově oddělená od backendové části a může být nasazena na jiném serveru než backend a může být psána v jiném jazyce než backendová část. Frontend je napsán v JavaScriptu díky jeho jednoduchosti a rychlosti vývoje.

První stránka, kterou člověk vidí, je přihlašovací formulář a navigační lišta. Po přihlášení se zobrazí stránka s úlohami, které může uživatel zapínat a vypínat. Dále je zde možnost zobrazit si informace o právě běžících úlohách a o všech dostupných úlohách.

Využití Reactu umožňuje stránce být tzv. SPA (Single Page Application) a tím pádem se nemusí stránka znovu načítat při každém přechodu mezi stránkami, což zvyšuje rychlost a plynulost stránky. Zároveň díky tomuto přístupu může webový server frontendu pouze posílat statické soubory a nemusí se starat o žádnou logiku, což zvyšuje bezpečnost a snižuje nároky na server.

3 Administrátorská Sekce

Sekce pro správce v tuto chvíli obsahuje tři části - vytváření nových úloh, vytváření vlajek, aktualizace uživatelů. Prostředí v administrátorské sekci je děláno tak, aby bylo intuitivní a nebyl problém s tímhle prostředím pracovat.

Vytváření nových úloh je děláno tak, že je nutné zadat pouze název úlohy, image úlohy a kategorii úlohy - veškeré ostatní části JSON manifestu úlohy jsou generovány na straně serveru automaticky, čímž se minimalizuje prostor na bezpečnostní chyby - není potřeba kontrolovat validitu odevzdaného JSON souboru, ale pouze těchto tří částí.

Vytváření vlajek pouze požaduje identifikační číslo ulohy, ke které se vlajka váže, a vlajku samotnou.

Aktualizace uživatelů umožňuje změnit uživateli práva nebo změnit heslo uživatele.

Pro bezpečnost této části je využíváno parametru "admin"u JWT tokenu, který je vytvořen při přihlášení uživatele s právy administrátora. Díky tomuto je možné jednoduše ověřit, zda je uživatel oprávněn k použití této části vzhledem k tomu, že s JSON web tokeny nelze manipulovat bez privátního klíče.

4 Access Box

Jakožto přístup k samotným úlohám je uživateli standardně poskytnut na žádost tzv. Access box, což je kontejner s image kalilinux/kali-last-release, který se nachází ve stejném namespace a tudíž ve stejné síti jakožto úlohy uživatele.

Toto řešení má oproti řešení pomocí VPN výhodu v tom, že není nutno instalovat žádný software na straně uživatele a není nutno vytvářet certifikáty, popř. uživatele, tudíž je mnohem jednodušší na implementaci a jednodušší na škálování jelikož lze tento kontejner spustit na každém nodu v Kubernetes clusteru.

Použití Kali Linuxu je z důvodu, že je to jedna z nejznámějších distribucí pro pentesting a je také jedna z nejvíce používaných distribucí pro tento účel, což umožňuje uživateli si toto prostředí vyzkoušet.

Ke kontejneru lze přistoupit z webového prostředí, kde lze vidět terminál realizovaný pomocí REST API.

ZÁVĚR

Cílem práce je webová aplikace a REST API pro práci s CTF systémem postaveným na platformě Kubernetes.

Aplikace je zálohavaná na GitHubu na adrese https://github.com/jan1s2-maturita