

# Technická dokumentace

---

## Spuštění pomocí docker compose

### Požadavky

- git
- docker

### Instalace

1. Nejdříve je potřeba naklonovat repozitář s aplikací: `git clone https://github.com/WebarchivCZ/linkra.git`
2. Poté se přesuneme do adresáře s aplikací `cd linkra`
3. Zde se v adresáři `docker` nacházejí podadresáře s připravenými docker-compose.yaml soubory pro různá prostředí, např. `docker/dev/docker-compose.yaml`
4. Ve vybraném docker-compose.yaml souboru je potřeba nastavit proměnnou `SERVER_HOST`. Nastavíme ji na URL, ze které bude aplikace dostupná (je potřeba zadat celou URL i s protokolem a případně portem, např. `https://linkra.webarchiv.cz`, nebo `http://10.10.10.10:8080`). Hodnota této proměnné musí odpovídat tomu, jak bude aplikace veřejně dostupná, tedy pokud bude dostupná přes reverzní proxy, měla by obsahovat adresu této reverzní proxy.
5. V elementu `volumes` změníme cestu k souboru s SQLite databází, která bude přimontovaná do kontejneru. Např. `jmeno_databaze.db:/mnt/storage.db:rw`. Uvedený soubor musí existovat a musí mít povolený zápis, jinak dojde k chybě. Soubor může být prázdný.
6. Dále je potřeba nastavit cestu pro ukládání archivních souborů z workeru. V compose souboru v elementu služby linkra-worker je potřeba do elementu `volumes` přidat nastavení pro mount adresáře, kam se mají soubory ukládat. Např. `/mnt/archiv/captures:/app/captures:rw`
7. Nakonec je potřeba se ujistit, že ve `worker-config.json` je `discardArchiveFiles` nastaveno na `false`. V případě, že si nepřejete ukládat archivní soubory (například při testovacím provozu), tak je možné nastavit na `true`, poté není třeba zadávat cestu pro archivní soubory, protože jí bude worker ignorovat.
8. Teď už je možné spustit aplikaci pomocí příkazu `docker compose up`. Např. `docker compose -f ./docker/zvolené_prostředí/docker-compose.yaml -p linkra up -d`

### Poznámky

- Pro produkční provoz se počítá se směrováním dotazů na aplikaci skrze reverzní proxy, která umožní použití https a rate limiting. Pro lokální provoz mimo internet není proxy nutná.
- Aplikaci nasazenou pomocí docker compose je možné snadno aktualizovat pomocí následujícího postupu:

```
# V kořenu repozitáře
# Stáhneme změny z githubu
git pull
# Vytvoříme a spustíme nové kontejnery
```

```
docker compose -f ./docker/zvolené_prostředí/docker-compose.yaml -p linkra up -d -  
-build --force-recreate
```

## Spuštění jako nativní proces (pro vývoj)

### Požadavky

- linux
- git
- go 1.24.0 nebo novější
- nodejs 22.20 (aktuálně bug v jedné závislosti neumožňuje použít vyšší verzi)
- npm
- valkey

### Instalace

1. Nejprve je potřeba nainstalovat požadavky. Konkrétně **go**, **nodejs** a **valkey** (alternativně by mělo být možné použít redis). Použijte oficiální postupy pro jejich instalaci. V případě aplikace valkey je možné použít libovolnou variantu (spustit jako příkaz nebo pomocí systemd nebo dockeru).
2. Naklonujeme repozitář s aplikací: `git clone https://github.com/WebarchivCZ/linkra.git`
3. Pokud ještě nemáme běžící instanci valkey, je potřeba ji spustit dříve, než se pokusíme spustit server.
4. Pokud jde o vývoj a dostačuje aplikace běžící na lokálním zařízení, stačí spustit `go run .` v kořeni repozitáře. Tento příkaz stáhne potřebné závislosti, zkompiluje a spustí server. Ve výstupu logu uvidíme adresu, na které bude aplikace dostupná.
5. Nyní, pokud chceme spustit i worker, tak přejdeme do adresáře `workers/scoop-worker`.
6. Zde nejprve spustíme příkaz `npm install`, který stáhne část závislostí pro worker.
7. Poté je potřeba ještě spustit `npm playwright install-deps chromium` pro doinstalování některých závislostí playwright.
8. Následně bude možné spustit worker pomocí `node main.js run`. Je možné příkaz opakovat a spustit více instancí worker.

### Poznámky

- Sever a worker mohou pracovat samostatně, veškerá komunikace mezi nimi probíhá skrze valkey.
- Server se dá pomocí příkazu `go build` (v kořeni repozitáře) zkompilovat do spustitelného souboru.
- Více workerů může běžet současně. Není k tomu potřeba žádná speciální konfigurace, pouze spuštění více worker procesů.

### Nastavení serveru

Nastavení serveru se mění pomocí proměnných prostředí. Pokud některá hodnota není explicitně nastavená, bude použita výchozí hodnota uvedená v závorce.

- **DB\_PATH** - Cesta k souboru SQLite databáze, kam se ukládají perzistentní data aplikace. (výchozí: `storage.db`)
- **VALKEY\_ADDR** - Adresa, na které běží valkey databáze. (výchozí: `localhost`)
- **VALKEY\_PORT** - Port, na kterém běží valkey databáze. (výchozí: `6379`)
- **SERVER\_ADDRESS** - Adresa a port, na kterém běží webové rozhraní serveru. (výchozí: `localhost:8080`)

- **SERVER\_HOST** - Protokol, adresa a host, na kterém je webové rozhraní dostupné z internetu. Může být stejné jako **SERVER\_ADDRESS**, pokud je vyžadován jen lokální přístup. (výchozí: `http://localhost:8080`)

## Nastavení workerů

Nastavení workerů se provádí pomocí JSON souboru. Vzorový soubor se nachází v `workers/scoop-worker/config.json`.

Musí to být JSON objekt, který obsahuje následující klíče:

- **discardArchiveFiles** - Boolean. Pokud je nastaven na `true`, tak worker přeskočí ukládání archivních souborů. Vhodné při testování, pokud nás zajímají vrácená metadata sklizně, ale nechceme vytvářet zbytečné soubory.
- **outputDir** - String. Cesta k adresáři, kam má worker ukládat archivní soubory.
- **valkeyUrl** - String. Adresa a port valkey databáze.
- **captureSettings** - Objekt, který může obsahovat konfiguraci pro Scoop. Pokud bude prázdný, tak budou použité rozumné výchozí hodnoty.

## Popis aplikace

Aplikace se skládá ze tří částí:

- **server** - poskytuje frontend, spravuje SQLite databázi, zařazuje zdroje ke sklizení, poskytuje přesměrování na archivní kopie
- **worker** - sklízí zadané zdroje, získává metadata ze sklizených dat, ukládá sklizená data
- **queue (fronta)** - komunikace mezi komponentami aplikace

### Server

Server poskytuje uživatelské rozhraní a přijímá požadavky na archivaci URL adres. Protože samotné adresy mohou být archivované více než jednou, je každé adrese přidělené jedinečné ID, které poté slouží k referenci dané URL adresy z webového rozhraní (např. při generování zkrácené archivní URL, nebo při zobrazení detailu), ale také při výměně metadat mezi workerem a serverem.

Po zpracování a zaznamenání URL do databáze je zařazena do fronty, odkud bude odebrána workerem ke sklizení a dalšímu zpracování. Poté, co worker dokončí práci, obdrží server odpověď s metadaty, která použije k vytvoření adresy archivní kopie ve waybacku (aplikaci k zobrazování archivních kopií webových stránek). Tento krok vytvoření archivní adresy je možné provést i v případě, že archivní kopie není zatím ve waybacku dostupná (např. z důvodu čekání na indexaci dat). Vygenerovaná archivní adresa odpovídá formátu nejběžnějších zpřístupňovacích aplikací používaných webovými archivy (Openwayback a PyWayback).

Server také pro každou URL určenou k archivaci vytvoří zkrácený link, který od chvíle, kdy dojde k úspěšnému sklizení URL, bude přesměrovávat na archivní kopii dané stránky ve waybacku webového archivu.

Server dále poskytuje rozhraní pro generování citací z archivovaných URL s možností předvyplnění některých informací pro vytvoření citace.

### Worker

Worker čte z fronty požadavky na sklizení URL adres. Aktuálně je implementován jako nodejs script s použitím sklízeče Scoop. Nástroj Scoop oproti např. sklízeči Heritrix, který je obvykle webovými archivy používán, umožňuje rychlé sklizení jedné URL, což umožňuje urychlené generování archivních adres, protože není potřeba čekat na zaindexování všech dat.

Worker je dále zodpovědný za extrakci metadat ze sklizených dat. Script otevře a zpracuje vygenerovaný WACZ soubor a odešle metadata potřebná pro generování archivní adresy zpátky do fronty, odkud si je převezme server. Worker poté uloží archivní data do specifikované cesty.

## Fronta

Server a worker komunikují pomocí fronty, která je aktuálně implementovaná pomocí Valkey (fork Redisu). Tato implementace umožňuje spustit více workerů, potenciálně i na více strojích a urychlit tak získávání archivních dat.