**STŘEDNÍ PRŮMYSLOVÁ ŠKOLA BRNO, PurkYŇOVA, příspěvková Organizace**



Realizace webhostingu na linuxové distribuci – Manuál pro administrátory

Karel Bašta

V4D

**Profilová část maturitní zkoušky**

**MATURITNÍ PRÁCE**

**BRNO 2022**

# Obsah

[Obsah 2](#_Toc101387268)

[Seznam obrázků 5](#_Toc101387269)

[1 Úvod 6](#_Toc101387270)

[2 Systémové požadavky 7](#_Toc101387271)

[3 Konfigurace použitých technologií 8](#_Toc101387272)

[3.1 Balíčkový manažer apt 8](#_Toc101387273)

[3.1.1 Konfigurace repositářů pro apt 8](#_Toc101387274)

[3.1.2 Nastavení preference mezi repositáři 9](#_Toc101387275)

[3.1.3 Aktualizace balíčků pomocí apt 10](#_Toc101387276)

[3.1.4 Instalace balíčků pomocí apt 10](#_Toc101387277)

[3.1.5 Odstranění balíčků pomocí apt 11](#_Toc101387278)

[3.1.6 Odstranění již nepotřebných knihoven 11](#_Toc101387279)

[3.1.7 Přeinstalace balíčků 11](#_Toc101387280)

[3.2 Démon systemd 12](#_Toc101387281)

[3.2.1 Nastavení procesů pro systemd 12](#_Toc101387282)

[3.2.2 Nastavení časovačů pro systemd 14](#_Toc101387283)

[3.2.3 Spuštění procesů 14](#_Toc101387284)

[3.2.4 Stav procesů 15](#_Toc101387285)

[3.2.5 Vypnutí procesů 15](#_Toc101387286)

[3.2.6 Restart procesů 15](#_Toc101387287)

[3.3 UFW 16](#_Toc101387288)

[3.3.1 Nastavení výchozích pravidel 16](#_Toc101387289)

[3.3.2 Udělení výjimky jednotlivým portům 17](#_Toc101387290)

[3.3.3 Aktivace a deaktivace UFW 17](#_Toc101387291)

[3.3.4 Restart UFW 17](#_Toc101387292)

[3.3.5 Výpis stavu UFW a udělených výjimek 17](#_Toc101387293)

[3.3.6 Smazání výjimky 18](#_Toc101387294)

[3.4 Fail2ban 18](#_Toc101387295)

[3.5 SSH server 18](#_Toc101387296)

[3.5.1 RSA klíče 19](#_Toc101387297)

[3.6 Webový s reverzní proxy server NGINX 20](#_Toc101387298)

[3.6.1 Představení konfiguračního souboru 20](#_Toc101387299)

[3.6.2 Začátek konfigurace 21](#_Toc101387300)

[3.6.3 Konfigurace zpracování požadavků 21](#_Toc101387301)

[3.6.4 Základní konfigurace HTTP a HTTPS serveru 22](#_Toc101387302)

[3.6.5 Konfigurace monitoringu provozu 23](#_Toc101387303)

[3.6.6 Konfigurace MIME typů 24](#_Toc101387304)

[3.6.7 Konfigurace paměti cache 24](#_Toc101387305)

[3.6.8 Přídavné konfigurace 24](#_Toc101387306)

[3.6.9 Konfgurace Brotli 25](#_Toc101387307)

[3.6.10 Konfigurace Pagespeed 25](#_Toc101387308)

[3.6.11 Konfigurace SSL 26](#_Toc101387309)

[3.6.12 Proces NGINX 31](#_Toc101387310)

[3.7 Databázový server MariaDB 31](#_Toc101387311)

[3.7.1 Konfigurace MariaDB 31](#_Toc101387312)

[3.7.2 Vytvoření nové databáze 32](#_Toc101387313)

[3.7.3 Vytvoření nového uživatele databáze 32](#_Toc101387314)

[3.7.4 Přidání práv pro správu databáze uživateli 33](#_Toc101387315)

[3.7.5 Zápis změn v právech uživatelů 33](#_Toc101387316)

[3.7.6 Proces MariaDB 33](#_Toc101387317)

[3.8 PHP 33](#_Toc101387318)

[3.8.1 Konfigurace PHP 33](#_Toc101387319)

[3.8.2 Proces php-fpm 34](#_Toc101387320)

[3.9 Cockpit 34](#_Toc101387321)

[3.10 Nextcloud 35](#_Toc101387322)

[3.10.1 Konfigurační soubor Nextcloud 35](#_Toc101387323)

[3.10.2 Proces Nextcloud 35](#_Toc101387324)

[3.10.3 Pravidelná žádost pro aktualizaci dat 36](#_Toc101387325)

[3.11 PCMS 36](#_Toc101387326)

[3.11.1 Databáze 36](#_Toc101387327)

[3.11.2 Konfigurační soubor 37](#_Toc101387328)

[3.11.3 Proces systemd 37](#_Toc101387329)

[4 Oficiální dokumentace balíčků 38](#_Toc101387330)

# Seznam obrázků

[Obrázek 1: Znázornění výměny klíčů 27](file:///C:\Users\basta\Documents\Personal\School\graduation\Thesis\Jeff\Manuál%20pro%20administrátory.docx#_Toc97154192)

# Úvod

Webhosting na linuxové distribuci nabízí a umožňuje velmi široké spektrum konfigurace. Tento manuál slouží administrátorům k základnímu porozumění jak konfiguraci jednotlivých balíčků, tak jejímu použití na tomto stroji.

Předem bych chtěl podotknout, že obsáhlost veškeré možné konfigurace všech použitích balíčků a programů je natolik velká, že zde bude vysvětlena pouze konfigurace použitá a její funkce. Ke každé z těchto konfigurací bude ovšem odkaz na dokumentaci daného balíčku či programu, kde bude možné se dozvědět o dalších možnostech.

# Systémové požadavky

Hardwarové požadavku pro možný běh tohoto počítače jsou přímo úměrné výkonu poskytnutému tomuto počítači plus požadavky samotného programu pro spuštění virtuálního počítače. Požadavky programů z řady VMware lze naleznout na následujícím odkaze: [https://www.vmware.com](https://www.vmware.com/resources/compatibility/search.php)

Mezi doporučenou konfiguraci výkonu virtuálního počítače patří:

|  |  |
| --- | --- |
| CPU | 2 až 4 jádra |
| RAM | 4 GB |
| Velikost úložiště | 20 GB + Potřebné úložiště pro cloudové úložiště dat pro uživatele |
| Síťový adaptér | V konfiguraci síťového mostu |

Tato konfigurace by měla vystačit pro běžné použití stroje při malé či středně velké zátěži. Pro využití ve větších firmách je vhodno buďto zátěž vyvážit mezi více takových strojů či zvýšit výkon počítače.

# Konfigurace použitých technologií

## Balíčkový manažer apt

### Konfigurace repositářů pro apt

Konfigurační soubory byly vytvářeny pomocí skriptů poskytnutých od vývojářů použitých balíčků. Tyto soubory je možné najít v adresáři /etc/apt/sources.list.d/. Každý ze souborů má následující strukturu:

deb [Filtr] adresa\_repositáře verze\_debian komponent

deb-src [Filtr] adresa\_repositáře verze\_debian komponent

Deb a deb-src indikují typ archivu. Deb znamená archiv již postavených binárních balíčků mezitím co deb-src označuje zdrojové kódy, ze kterých je možno balíček samostatně postavit.

Dále v sekci filtr je možné specifikovat například architekturu procesoru, pro které si přejeme balíčky instalovat pomocí

[arch=amd64,arm64]

Nebo je možné specifikovat cestu k GPG klíči, pod kterým budou muset být balíčky podepsány pro zvýšení bezpečnosti.

[signed-by=/usr/share/keyrings/nodesource.gpg]

Do sekce adresy repositáře je potřeba zadat URL adresu daného repositáře, například:

<https://dlm.mariadb.com/repo/mariadb-server/10.7/repo/debian>

<https://packages.sury.org/php/>

<https://deb.nodesource.com/node_16.x>

Další je nutno specifikovat verzi Debianu, na kterém se budou balíčky instalovat. V našem případě se jednalo o verzi „bullseye“, v době instalace existoval též Debian verze „bookworm“ a špičková verze „sid“.

Mezi komponenty se určují licence balíčků buďto „main“ pro balíčky splňující zásady [DFSG](https://www.debian.org/social_contract.cs.html) nebo „contrib“ pro DFSG balíčky, které jsou vyžadovány „non-free“ balíčky. „non-free“ balíčky nesplňují zásady DFSG.

### Nastavení preference mezi repositáři

Apt obsahuje ještě další konfigurační soubory například v adresáři /etc/apt/preferences.d/, ve kterém byl také pomocí skriptu vytvořen soubor nginx-block.conf. Soubory v tomto adresáři mají následující strukturu:

Package: název\_balíčku

Pin: filtr

Pin-Priority: priorita

Účel těchto souborů je určení priority pro zdroje poskytující stejný balíček. Jestliže balíček „php“ poskytuje výchozí Debian repositář a zároveň námi zvolený repositář, pak je vhodné vytvořit konfiguraci s preferencí námi zvoleného repositáře.

Do sekce „Package:“ se určuje název balíčku či balíčků, ke kterému chceme prioritně určit zdroj.

Package: nginx\*

Do sekce „Pin:“ se určuje samotný zdroj pro který určujeme prioritu a do sekce „Pin-Priority“ se určuje samotná priorita určená číslem, kde jeho hodnota má jiný význam podle následujících pravidel a poté podle většího čísla:

|  |  |
| --- | --- |
| <0 | Způsobí úplný zákaz instalace balíčků |
| 0–99 | Nainstaluje balíček, jestliže není dostupná žádná jiná verze |
| 100-499 | Nainstaluje balíček, jestliže není dostupná novější verze |
| 500-999 | Nainstaluje balíček, jestliže není dostupná novější verze ve výchozím Debian repositáři |
| >1000 | Nainstaluje balíček, i když by měl snížit jeho verzi |

### Aktualizace balíčků pomocí apt

Pro veškerou úpravu balíčků je nutno použít příkaz sudo, který umožňuje provádět příkazy se zvýšeným oprávněním.

Pro aktualizaci balíčků spravovaných pomocí manažera apt je nutno nejdříve aktualizovat databázi balíčků z nastavených repositářů pomocí příkazu:

sudo apt update

Poté je možné teprve stáhnout nejnovější verze balíčků pomocí příkazu:

sudo apt upgrade

Po použití tohoto příkazu manažer vypíše seznam balíčků k aktualizaci společně s detaily o velikosti přenesených dat, které je nutno potvrdit. Jestliže by tento krok měl být přeskočen, je možné využít příznak „-y“.

Jestliže se chce aktualizace pouze stáhnout, ale ještě neinstalovat, je možné využít příkaz:

sudo apt upgrade –-download-only

Před instalací jakéhokoli nového balíčků je vhodné nejdříve aktualizovat dosud instalované balíčky či aspoň aktualizovat databázi, aby byla zaručena integrita balíčků.

### Instalace balíčků pomocí apt

Před instalací balíčků je vhodné nejdříve nainstalovat aktualizace balíčků již nainstalovaných, viz předchozí kapitola.

Pro instalaci nových balíčků je vhodné použití následující příkaz:

sudo apt install htop vim git

Po zvolení konkrétní funkce programu apt je možné zapsat jeden či více názvů balíčků, se kterými chceme pracovat.

### Odstranění balíčků pomocí apt

Pro odinstalovaní balíčků je potřeba zvolit příkaz:

sudo apt remove htop

Tento příkaz daný balíček odinstaluje, ale ponechá jeho instalační soubory v počítači. Pro odstranění těchto souborů je nutno buďto dodat příznak „--purge“ či zaměnit příkaz remove za příkaz purge.

### Odstranění již nepotřebných knihoven

Jestliže nastane situace, kde žádný balíček již nepotřebuje nainstalovanou knihovnu je možné tyto knihovny odstranit pomocí příkazu:

sudo apt autoremove

### Přeinstalace balíčků

Jestliže dojde u některého z balíčků k problému, u kterého je řešením přeinstalace daného balíčku je možné využít příkaz:

sudo apt reinstall htop

## Démon systemd

### Nastavení procesů pro systemd

Ručně vytvořené konfigurační soubory procesů pro démona systemd byly ukládány do adresáře /etc/systemd/system/. Soubory mají koncovku „.service“ a využívají takzvanou INI strukturu, která dělí konfiguraci na skupiny, ke kterým jsou přiřazeny názvy proměnných a jejich hodnoty. Příkladná ukázka takové konfigurace:

[Unit]

Description=Popis\_procesu

[Service]

Type=Typ\_procesu

User=Uživatel

Group=Skupina\_uživatelů

Environment=Proměnná=Hodnota

WorkingDirectory=Adresář\_procesu

ExecStart=Příkaz\_pro\_spuštění

Restart=Pravidlo\_pro\_restrt

[Install]

WantedBy=multi-user.target

Tato konfigurace se dělí na tři části.

První část „Unit“ je pro metadata daného procesu. Zde je možno určit jméno procesu, cestu k jeho dokumentaci, jeho závislost vůči ostatním procesům (Například spuštění procesu po připojení k internetu, který je v počítači veden jako proces) a další... V naší konfiguraci se v této části se zde objevuje akorát popis procesu.

Druhá část „Service“ je určena pro samotnou konfiguraci procesu a objevuje se spoustu důležitých nastavení. Zde je význam vybraných důležitých nastavení:

* **Type** – Určuje typ procesu v zavilosti na nakládání s jeho podprocesy. Mezi základní hodnoty této proměnné patří:
  + „simple“ pro procesy s jedním procesem bez speciálních pravidel.
  + „forking“ pro procesy vytvářející podprocesy kde se hlavní proces ukončí po spouštění všech podprocesů. Pro takové procesy je důležité toto specifikovat, jinak by systemd prohlásil proces za ukončený po ukončení mateřského procesu.
  + „oneshot“ pro pravidelně volané rychlé procesy
* **User** – Určuje, za jakého uživatele se má proces spustit. Důležité kvůli právům pro zápis a čtení souborů se kterými bude proces pracovat
* **Environment** – Určuje hodnotu proměnných pro prostředí, ve kterém se bude proces spouštět. Některé procesy tyto proměnné využívají místo příznaků
* **WorkingDirectory** – Určuje, v jakém adresáři bude proces spouštět.
* **ExecStart/ExecStartPre/ExecStartPost/ExecReload/ExecStop/ExecStopPost**
  + Všechna tato nastavení určují příkaz, který se má spouštět při události popsané anglicky v názvu těchto nastavení
  + **ExecStart** je z těchto nastavení nejdůležitější, jelikož určuje, jakým příkazem se bude proces spouštět
* **Restart** a **RestartSec** – Nastavení určující, v jakém případě se má proces spustit znovu
  + **Restart** – Možné hodnoty jsou „always“, „on-failure“, „on-success“ a další… Hodnoty znamenají, že se restart provede pouze podle anglického překladu těchto hodnot
  + **RestartSec** – Jak dlouhou dobu se má čekat, než se provede restart

Poslední, ala nepovinná část tohoto konfiguračního souboru je „Install“, ve které se určuje také určují vztahy s jinými procesy, ovšem v této části se ovlivňuje nejenom tento proces, ale i další specifikované procesy.

### Nastavení časovačů pro systemd

Pro pravidelné vykonání Nextcloud procesu bylo nutné vytvořit časovač, který volal jeho samotný proces. Časovače se nachází ve stejné složce jako procesy, mají koncovku „.timer“ a mají následující strukturu:

[Unit]

Description=Run Nextcloud cron.php every 5 minutes

[Timer]

OnBootSec=5min

OnUnitActiveSec=5min

Unit=nextcloudcron.service

[Install]

WantedBy=timers.target

Zde můžeme vidět nějaké nové nastavení mezi která patří:

* **OnBootSec** – Definuje, jak dlouho po startu počítače se má čekat před prvním voláním časovače
* **OnUnitActiveSec** – Definuje, jak dlouho se má čekat mezi voláními časovače
* **Unit** – Definuje, jaký proces se má pomocí tohoto časovače volat

Ostatní nastavení mají stejný význak jako u procesů samotných

### Spuštění procesů

Po každé úpravě souborů procesů či časovačů je nutné tyto změny zavést do samotného programu systemd pomocí následujícího příkazu:

sudo systemctl daemon-reload

Procesy se jednorázově spouští pomocí následujícího příkazu:

sudo systemctl start název\_procesu

Jestliže mají například proces i časovač stejné jméno, je nutné u volání specifikovat koncovku buďto „.service“ či „.timer“. Bez specifikované koncovky systemd autmoaticky pracuje se souborem s koncovkou „..service“.

Pro automatické spuštění procesu při startu počítače je možné použít následující příkaz:

sudo systemctl enable název\_procesu

### Stav procesů

Pro zjištění stavu procesu je možné využít následující příkaz:

sudo systemctl status název\_procesu

### Vypnutí procesů

Pro vypnutí procesu, dokud není vypnut počítač je možné využít následující příkaz:

sudo systemctl stop název\_procesu

Jestliže se přeje proces již nespouštět při startu počítače je možné využít příkaz:

sudo systemctl disable název\_procesu

### Restart procesů

Pro běžný restart procesu je možné použít příkaz:

sudo systemctl restart název\_procesu

## UFW

Konfigurační soubory balíčku UFW se nachází na cestě /etc/ufw/. V tomto adresáři se nachází konfigurace, jakým způsobem UFW nastavuje [iptables](https://linux.die.net/man/8/iptables) pravidla. Tato pravidla nebyla změněna. Dále se zde nachází soubor ufw.conf, ve kterém se určuje, jestli se má UFW spouštět při startu počítače a jaká má být úroveň záznamů tohoto programu do systémových logů.

UFW dále má jeden konfigurační soubor na cestě /etc/default/ufw, ve kterém se určují výchozí pravidla, jestliže nejsou specifikována jinak. Dále se zde určuje užití IPv6 a moduly pro UFW. V této konfiguraci také nebyla učiněna žádná změna.

Veškeré změny v konfiguraci byly učiněny pomocí následujících příkazů:

sudo ufw default deny incoming

sudo ufw default allow outgoing

sudo ufw allow 80/tcp

sudo ufw allow 443

sudo ufw allow 25522/tcp

První dva příkazy nastavili výchozí zákaz všech připojení směrem do zařízení a povolení všech připojení směrem ze zařízení. Další příkazy povolili komunikaci po obou směrech pro specifikované porty a protokoly. Mezi těmito porty byly porty 80 a 443 pro možnost přístupu k webovým stránkám a port 25522, na kterém byla nastaven vzdálený přístup.

### Nastavení výchozích pravidel

Pro nastavení výchozích pravidel pro komunikaci je možné využít následující příkaz:

sudo ufw default allow/deny incoming/outgoing

Hodnota „allow“ komunikaci umožňuje a „deny“ ji zakazuje. Hodnota „incoming“ znamená příchozí a „outgoing“ znamená odchozí připojení.

### Udělení výjimky jednotlivým portům

Pomocí následujícího příkazu je možné udělit výjimku pro komunikaci specifikovanému portu s možností specifikace protokolu. Jestliže protokol není specifikován, je pravidlo aplikováno na oba protokoly.

sudo ufw pravidlo port/protokol

Mezi možná pravidla opět patří hodnoty „allow“ pro umožnění komunikace a „deny“ pro její zákaz. Další možné hodnoty jsou „reject“ a „limit“. Hodnota „reject“ pošle zdroji požadavku informaci o chybě. Hodnota „limit“ značí zákaz komunikace po šesti požadavcích v rozmezí 30 sekund. Tato hodnota nelze změnit.

### Aktivace a deaktivace UFW

Pomocí následujícího příkazu je možné balíček UFW aktivovat. Je příhodné mít minimálně udělenou výjimku pro vzdálený přístup, jelikož by aktivace znamenala jeho přerušení.

sudo ufw enable

Deaktivace UFW se poté provádí pomocí příkazu

sudo ufw disable

### Restart UFW

Po zavedení nových pravidel je vhodné UFW restartovat. K tomu je možné využít příkaz:

sudo ufw reset

### Výpis stavu UFW a udělených výjimek

Pomocí následujícího příkazu je možné vypsat stav UFW a jestliže je UFW aktivní, tak se vypíší i aktivní výjimky.

sudo ufw status

### Smazání výjimky

Pomocí indexu vypsané výjimky či názvu je možné výjimku odstranit pomocí následujícího příkazu:

sudo ufw delete index/název

## Fail2ban

Fail2ban nebyl po instalaci nijak konfigurován. Výchozí funkce postačuje pro účely počítače.

Adresář konfiguračních souborů této aplikace je na cestě /ect/fail2ban. V tomto adresáři se nachází konfigurační soubory s koncovkou „.conf”. Všechnu konfiguraci v nich je možno přepsat pomocí stejnojmenných souborů s koncovkou „.local“. Pro změnu konfigurace se doporučuje využívat těchto „.local“ souborů, jelikož se výchozí soubory přepisují s každou aktualizací balíčku.

Mezi důležité konfigurační soubory patří soubor fail2ban.conf, ve kterém je veškerá základní konfigurace balíčku. Mezi tuto konfiguraci patří nastavení databáze tohoto programu, záznam procesu aj.

IP adresy se zakazují dle pravidel specifikovaných souborech ve složce „filter.d“.

Zakázané IP adresy se zavírají do tzv. „vězení“, která se konfigurují v souboru „jail.conf“

Konfigurace tohoto balíčku je velmi komplexní, a proto doporučuji si přečíst oficiální dokumentaci ze stránek specifikovaných v poslední kapitole manuálu.

## SSH server

Konfigurační soubor pro SSH server se nachází na cestě /etc/ssh/sshd\_config. Při aktualizaci tohoto balíčku se balíčkový balíček zeptá zdali-li si přejeme konfiguraci ponechat nebo zaměnit za novou. Doporučeno je konfiguraci ponechat.

Tento soubor obsahuje konfiguraci taktéž v INI formátu, ovšem tentokrát bez skupin. Tudíž se zde píšou pouze proměnné, za kterými následuje jejich hodnota.

V tomto souboru byly provedeny dvě změny:

* Na konec souboru byl připsáno nastavení „PasswordAuthentication no“, které zakázalo přihlašování pomocí hesla, jelikož se využívá rsa-klíčů
* Na řádku 13 byla odstraněna

### RSA klíče

Tvoří se na počítači, ze kterého budeme chtít počítač ovládat pomocí příkazu:

ssh-keygen

Tento příkaz vygeneruje veřejný a soukromý klíč pomocí kterého bude možno provést autentizaci uživatele bez potřeby hesla. Vygenerovaný veřejný klíč najdeme nejčastěji na adrese ~/.ssh/id\_rsa.pub. Tento klíč se následně přenese na server klidně pomocí dočasného vystavení klíče na internet a následné stažení. Tento veřejný klíč není na internetu hrozbou, jelikož průměrná doba rozšifrování takového klíče je okolo 300 tisíc miliard let. Důležité je ovšem nevystavit internetu klíč soukromý. Rozdíl mezi klíči je v jejich hlavičce. V hlavičce soukromého klíče bude text:

-----BEGIN OPENSSH PRIVATE KEY-----

Mezitím co v hlavičce veřejného klíče bude:

ssh-rsa AAAA + zbytek klíče

Stažený veřejný klíč je nutno uložit do souboru ~/.ssh/authorized\_keys k danému uživateli, za kterého se budou moci lidé připojit.

## Webový s reverzní proxy server NGINX

### Představení konfiguračního souboru

Konfigurační soubor programu NGINX se nachází na cestě /etc/nginx/nginx.conf. Tento soubor byl vytvořen celý ručně pro maximální výkon a bezpečnost při zpracování žádosti a zasílání odpovědi. Tento soubor je velmi obsáhlý, tudíž zde postupně rozeberu jednotlivé části souboru:

Konfigurace NGINX se rozděluje na 5 částí:

* „main“ – Hlavní nastavení procesů
* „events“ – Nastavení, jakým způsobem se zpracovávají požadavky mezi procesy
* „http“ – Nastavení zpracování HTTP a HTTPS požadavků
* „server“ – Nastavení virtuálních hostů
* „location“ – Nastavení odpovědí pro jednotlivé adresáře a soubory hosta

Jako host se v našem případě bere webové adresa obsahující doménu a subdoménu. Příklad:

* jeff.sspbrno.cz
* postman.sspbrno.cz
* cloud.sspbrno.cz

Na jednom počítače lze takto pomocí virtuálních hostů obsluhovat více takových webových adres. Jako soubor se v našem případě bere poslední položka požadované URL adresy. Příklad:

* cloud.sspbrno.cz/**index.php**
* sspbrno.cz/dokumenty/**sr.pdf**

Jako adresáře v našem případě bereme veškeré položky mezi cílovým souborem a doménou oddělené lomítky.

* [https://jeff.sspbrno.cz/**nextcloud**/**apps**/**dashboard**/](https://jeff.sspbrno.cz/nextcloud/apps/dashboard/)

### Začátek konfigurace

V hlavičce našeho konfiguračního souboru je možné vidět tyto parametry:

user www-data;

pid /run/nginx.pid;

Nastavení „user“ určuje, za kterého uživatele se má NGINX spustit. Je důležité, aby tento uživatel měl přístup ke všem potřebným souborům a soketům. „pid“ určuje cestu k souboru, ve kterém bude uloženo identifikační číslo procesu NGINX.

worker\_processes auto;

worker\_rlimit\_nofile 65535;

Nastavení „worker processes“ určuje počet procesů, které můžou zpracovávat požadavky. Ve většině případů je doporučeno mít stejný počet procesů jako vláken procesoru. Parametr „auto“ zaručí tento počet. Parametr „worker\_rlimit\_nofile“ určuje jaký je maximální počet otevřených souborů jedním procesem. Určeno dostatečným množstvím, ale zároveň omezeným pro zamezení zahlcení disku programem na hodnotu „65535“.

Poté se zde objevuje import konfiguračních souborů modulů, které se nemají vyskytovat v části http. V naší konfiguraci sice žádný takový nemáme.

# Load modules

include /etc/nginx/modules-enabled/\*.conf;

### Konfigurace zpracování požadavků

Poté se zde objevuje část „events“, ve které se objevuje tato konfigurace:

events {

worker\_connections 65535;

multi\_accept on;

use epoll;

epoll\_events 512;

}

Nastavení „worker connections“ určuje maximální počet připojení k jednomu procesu. Připojení jsou rozložena mezi procesy, tudíž bychom se neměli setkat s nemožností připojit další zařízení nebo přetížením pouze jednoho procesu. Nastavení „multi\_accept“ umožňuje zpracování více žádostí o připojení jedním procesem, bez této specifikace by se žádosti zpracovávali popořadě.

Nastavení „use“ určuje jakým způsobem se zpracovávají připojení v NGINX. Metoda „epoll“ je speciální metoda pro Linux verze 2.6 a vyšší zaručující nejvyšší efektivitu. Tato metoda umožňuje deklarace počtu komunikačních kanálů se vstupně výstupním kanálem počítače pomocí nastavení „epoll\_events“. Tyto kanály inicializují připojení klienta s NGINX pomocí takzvané události, anglicky eventu. Po zpracování připojení z epoll se připojení již provádí přímo do NGINX.

### Základní konfigurace HTTP a HTTPS serveru

Nyní se můžeme vrhnout na popis nastavení zpracování HTTP a HTTPS požadavků. Základní nastavení http bloku je následující:

http {

charset utf-8;

sendfile on;

aio threads;

tcp\_nopush on;

tcp\_nodelay on;

server\_tokens off;

log\_not\_found off;

types\_hash\_max\_size 2048;

types\_hash\_bucket\_size 64;

client\_max\_body\_size 16M;

Základní nastavení tohoto bloku zahrnuje nastavení „charset“ určující znakovou sadu posílané odpovědi. Znaková sada byla nastavena na „utf-8“, která zaručuje správné zobrazení všech znaků. Jestliže nebude znaková sada zdrojových data shodná, tak se převede.

Poté se zde definuje nastavení „sendfile“, které jestliže je nastavené na hodnotu „on“ umožňuje přímé zasílání souboru ze souborového systému počítače. Jinak by se soubor nejdříve zkopíroval do buffer paměti a až z něho by se odesílal. Zároveň s tímto se zde určuje nastavení „aio“, které umožňuje asynchronní práci se soubory. S hodnotou „threads“ umožňuje asynchronní práci s daty každému procesu.

Nastavení „tcp\_nopush“ umožňuje zaslání HTTP hlaviček společně s částí zasílaného souboru. Nastevení „tcp\_nodelay“ umožňuje zasílání souborů za pomoci jeho menších částí místo využívání jednoho velkého paketu pro celý soubor. Toto nastavení snižuje odezvu při žádosti o soubor.

Poté zde máme konfiguraci využitou pro změnu komunikace z HTTP protokolu na WebSocket protokol pomocí:

map $http\_upgrade $connection\_upgrade {

default upgrade;

'' close;

}

Tato změna se provádí pouze jestliže cílová aplikace určí a pomocí zde zmapované proměnné, která se poté využívá při přenosu žádosti na server dané aplikace.

### Konfigurace monitoringu provozu

Nyní zde máme řádek pro aktivaci monitoringu provozu v této oblasti za pomoci modulu vfs:

vhost\_traffic\_status\_zone;

### Konfigurace MIME typů

Následuje definice takzvaných „MIME“ typů, které můžeme poté přiřadit jednotlivým souborům zasílaným jako odpověď na požadavky, pomocí kterých internetový prohlížeč může optimalizovat práci s těmito typy souborů. Samotné definice jsou obsažené v souboru /etc/nginx/mime.types. Jako výchozí typ odpovědi byl zvolen „octet-steam“.

# MIME

include /etc/nginx/mime.types;

default\_type application/octet-stream;

### Konfigurace paměti cache

Pro zvýšení výkonu při zpracování častých požadavků byla nastavena paměť cache. Pomocí této paměti jsou odpovědi na tyto požadavky optimalizované nemusí se celá odpověď generovat znova.

# Cache

proxy\_cache\_path /var/cache/nginx levels=1:2 keys\_zone=default\_cache:10m max\_size=2g inactive=60m use\_temp\_path=off;

Tato cache pamět má své místo na cestě /var/cache/nginx. Paměť byla nastavena na zachytávání častých požadavků za posledních 10 minut. Maximální velikost této paměti byla nastavena na 2 GB a byla nastavena pro automatické čištění po 60 minutách neaktivity.

### Přídavné konfigurace

Před importem samotných virtuálních hostů se importují tři konfigurační soubory ze složky /etc/nginx/conf.d/. V této složce se nachází specializované konfigurace pro komprimaci odpovědí, optimalizací webových stránek a nastavení bezpečné SSL komunikace.

include /etc/nginx/conf.d/\*.conf;

### Konfgurace Brotli

Takto vypadá konfigurace pro komprimaci odpovědí pomocí modulu brotli.

# Brotli compression

brotli on;

brotli\_static on;

brotli\_comp\_level 6;

brotli\_types \*;

Nastavení „brotli“ určuje jeho použití. Bylo nastaveno na „on“ jako „zapnuto“. Nastavení „brotli\_static on“ umožnilo použití před-kompresovaných souborů pomocí algoritmu brotli. Nastavení „brotli\_comp\_level“ určuje hodnotu od 0 do 11 pro kvalitu komprese. Čím vyšší kvalita komprese, tím vyšší je zátěž na počítač a tím déle bude komprese trvat. Byl zvolen doporučený průměr v podobě čísla 6. Poté zde jsou ještě určeny, jaké MIME typy souborů se mají kompresovat pomocí nastavení „brotli\_typer“ s hodnotou „\*“ pro všechny typy.

### Konfigurace Pagespeed

Konfigurace pro optimalizaci webových stránek vypadá takto:

# Page speed

pagespeed on;

pagespeed FileCachePath /var/ngx\_pagespeed\_cache;

Nastavením „pagespeed on“ se aktivuje využití balíčku pagespeed, které v sobě obsahuje funkce pro zvýšení snížení odezvy a velikosti přenesených dat podle filtrů dostupných k nahlédnutí na adrese [https://www.modpagespeed.com/](https://www.modpagespeed.com/doc/filters). Tento balíček upravuje žádané soubory a jeho dočasná paměť těchto souborů se nachází na specifikované cestě z nastavení „pagespeed FileCachePath“.

### Konfigurace SSL

Poslední je trochu složitější konfigurace, která zabezpečuje správnou a bezpečnou komunikaci pomocí SSL:

# SSL

ssl\_prefer\_server\_ciphers on;

ssl\_session\_timeout 1d;

ssl\_session\_cache shared:SSL:10m;

ssl\_session\_tickets off;

ssl\_early\_data on;

# Diffie-Hellman parameter for DHE ciphersuites

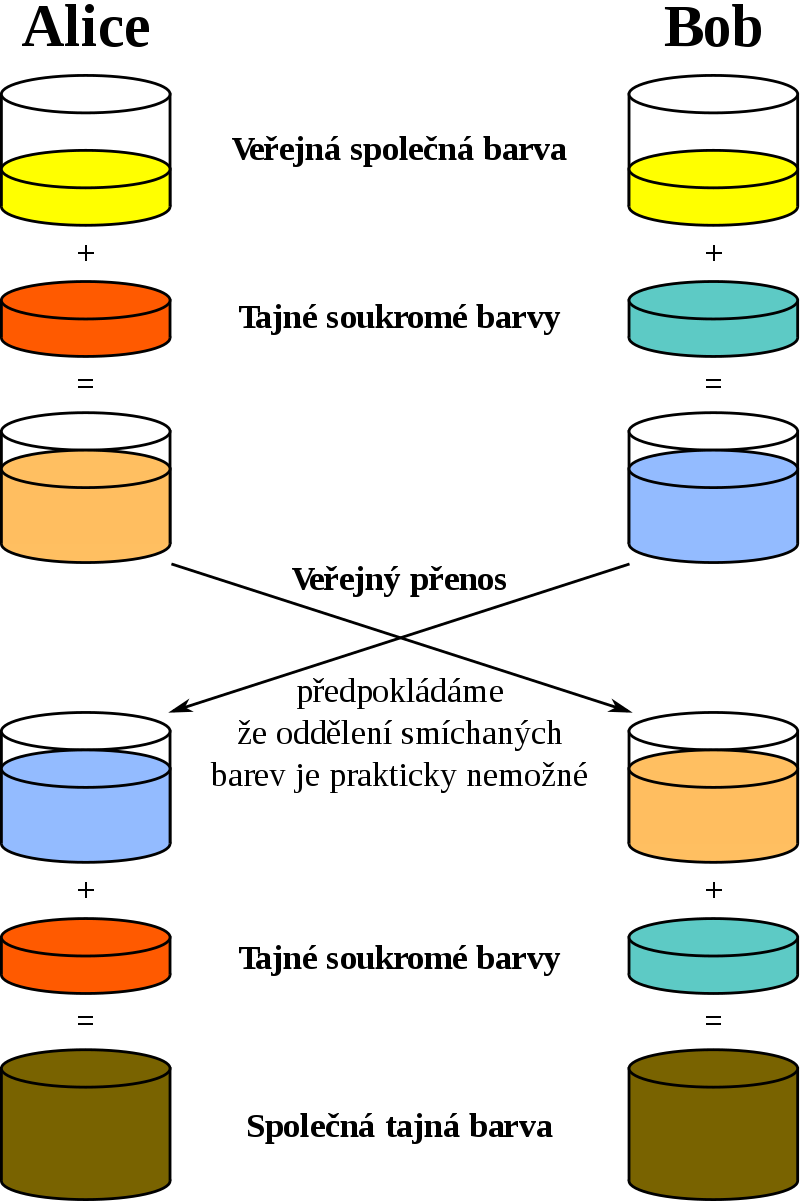
ssl\_dhparam /etc/nginx/dhparam.pem;

# Mozilla Intermediate configuration

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:…:DHE-RSA-AES256-GCM-SHA384;

Jako první nastavení pro SSL komunikaci je preference serverových klíčů pro šifrování komunikace „ssl\_prefer\_server\_ciphers on“. Nastavení „ssl\_session\_timeout“ určuje, jak dlouhou dobu může klient využívat poskytnuté parametry pro SSL komunikaci bez jejich obnovy, tato hodnota byla nastavena na jeden den. Dále je zde definovaná paměť pro SSL parametry komunikace použitelná pro server pomocí nastavení „ssl\_session\_cache“. Tato paměť je nastavena na maximální velikost 10 MB se jménem „SSL“. Hodnota „shared“ v tomto případě znamená, že je paměť použitelná všemi procesy NGINX. Dále se zde pomocí nastavení „ssl\_session\_tickets off“ zakazuje obnovení SSL parametrů pomocí TLS požadavků a pomocí „ssl\_early\_data on“ se umožňuje zaslání SSL dat při žádosti o komunikace místo až po ní.

Dále se v této konfiguraci vyskytuje cesta k Deffie-Hellman parametru. Tento parametr si můžeme představit jako veřejný klíč, pomocí kterého se vytváří soukromé klíče pro další komunikaci. Tento klíč byl stažen z [https://ssl-config.mozilla.org/](https://ssl-config.mozilla.org/ffdhe2048.txt).

Obrázek 1: Znázornění výměny klíčů

Klíč se také dá vygenerovat pomocí příkazu:

openssl dhparam -out jméno\_souboru velikost

Podle velikosti se poté určuje síla daného klíče. Funkce tohoto klíče se dá představit pomocí obrázku vpravo:

Po importu celé této konfigurace bylo možné importovat všechny virtuální hosty. V našem případě se jedná pouze o hosta s adresou „jeff.sspbrno.cz“.

include /etc/nginx/sites-enabled/\*.conf;

Konfigurace začíná definicí lokálních serverů daných aplikací od aplikace cockpit po pcms. Tato definice umožňuje jednoduché použití proměnné jako cíl žádosti:

# Upstreams

upstream cockpit{

server 127.0.0.1:9090;

}

upstream php80{

server unix:/run/php/php8.0-fpm.sock;

}

upstream pcms\_backend{

server unix:/tmp/pcms.socket;

}

upstream pcms\_admin{

server 127.0.0.1:25580;

}

Definice serverů měli následující strukturu:

upstream název\_serveru{

server adresa;

}

Jako adresu je možné určit IP adresu s portem dané aplikace či cestu k jeho soketu.

Konfigurace pokračuje definicí poslechu na portu 80 pro hosta „jeff.sspbrno.cz“. Na veškeré požadavky tohoto hosta bude odpovězeno zasláním HTTP kódu 301 se stejnou adresou, ale za použití protokolu HTTPS. Tato odpověď značí permanentní přesun stránek na jinou adresu. Zaručuje se tím robustní přenos komunikace z nezabezpečeného HTTP protokolu na zabezpečený protokol HTTPS. Protokol HTTPS poskytujeme stejným způsobem, ale na portu 443. Oba tyto porty jsou standardem pro tyto protokoly a každý internetový prohlížeč ví co má v tomto případě dělat.

# Redirect all http connections to https

server {

listen 80 default\_server;

listen [::]:80 default\_server;

server\_name jeff.ssbpbrno.cz;

return 301 https://$host$request\_uri;

}

# HTTPS server

server {

# HTTPS/2

listen 443 ssl http2;

listen [::]:443 ssl http2;

Využíváme zde i lepšího protokolu http2 pro komunikaci, který zaručuje vyšší rychlost přenosu pomocí možnost zaslání odpovědi postupně tak, aby se byla schopna stránka vykreslovat i při načítání. Tudíž není nutno čekat na načtení celého obsahu pro jeho zobrazení.

Zároveň na portu 443 posloucháme komunikaci pomocí HTTP3 stejným způsobem.

# HTTPS/3

listen 443 quic reuseport;

listen [::]:443 quic;

Poslouchání na portu je definováno pomocí „listen“ a čísla portu. Po čísle portu je možné specifikovat různé parametry od ssl pro šifrovanou komunikaci, pro kterou bude nutné následně importovat ssl certifikát nebo použití protokolu http2 či hodnota „default\_server“, která znamená, že veškeré požadavky, které nemají již specifikovaného virtuálního hosta na tomto serveru budou zpracovány následujícím hostem.

server\_name jeff.ssbpbrno.cz;

root /var/www;

Po definicí portů poslechu je nutné specifikovat na jaké doméně tento host pracuje. U nás se jednalo o doménu „jeff.sspbrno.cz“. Pomocí „root“ se definuje z jakého adresáře poskytujeme data pro požadavek na tuto doménu.

# Caching

proxy\_cache default\_cache;

Pro zrychlení odpovědí na časté dotazy byla využita dříve vytvořená paměť cache.

Důležitou součástí naší služby je záznam požadavků a úspěšnost odpovědí na ně. Tento záznam byl nastaven na zápis so souborů na cestě /var/log/nginx.

# Logging

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error.log warn;

Poté byli importovány SSL certifikát a klíč z definovaných cest.

# SSL

ssl\_certificate /var/cert/example.crt;

ssl\_certificate\_key /var/cert/example.key;

A nakonec se ke každé odpovědi přidali následující bezpečnostní hlavičky:

# Security

include snippets/security.conf;

Každá z těchto hlaviček má svůj vlastní význam pro prohlížeče. Vysvětlení všech těchto hlaviček je velmi dlouhé a mimo rozsah mé práce. Přehled nyní používaných HTTP hlaviček a jejich význam je možné najít na adrese [https://developer.mozilla.org/](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers).

# security headers

add\_header X-XSS-Protection "1; mode=block" always;

add\_header X-Content-Type-Options "nosniff" always;

add\_header Referrer-Policy "no-referrer-when-downgrade" always;

add\_header Content-Security-Policy "default-src 'self' http: https: data: blob: 'unsafe-inline' 'unsafe-eval'; frame-ancestors 'self';" always;

add\_header Permissions-Policy "interest-cohort=()" always;

add\_header Strict-Transport-Security "max-age=31536000" always;

add\_header X-Download-Options "noopen" always;

add\_header X-Frame-Options "SAMEORIGIN" always;

add\_header X-Permitted-Cross-Domain-Policies "none" always;

add\_header X-Robots-Tag "none" always;

add\_header X-Cache-Status $upstream\_cache\_status always;

Pro přenesení správných informací ohledně připojení mezi samotnou aplikací a uživatelem je nutné přenést aplikaci informace o uživateli pomocí následujících hlaviček:

# Proxy headers

proxy\_set\_header Host $host;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header X-Forwarded-Host $host;

proxy\_set\_header X-Forwarded-Server $host;

proxy\_set\_header X-Forwarded-Port $server\_port;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

proxy\_set\_header X-SSL-Session $ssl\_session\_id;

proxy\_set\_header X-Real-IP $remote\_addr;

Tyto hlavičky by se měli importovat před veškerou komunikací s aplikací. Každá aplikace využívá jiné hlavičky a je důležité si je zjistit před použitím Tyto hlavičky je zbytečné importovat při hostingu statických stránek. Import byl proveden pomocí:

# Proxy config and security

include snippets/proxy.conf;

Pro každou aplikaci byla dále definována vlastní pravidla dle doporučení poskytovatelů aplikací.

Zároveň byl definován pro každou aplikaci jiný soubor pro záznam logů pro jednodušší práci s nimi.

Pro další otázky ohledně konfigurace NGINX doporučuji navštívit oficiální dokumentaci.

### Proces NGINX

Proces NGINX se ovládá pomocí příkazů pro procesy systemd. Název procesu je „nginx.service“. Pro syntaxi příkazů viz [Démon systemd](#_Démon_systemd).

Pro zjištění validity konfigurace před jejím použitím je možné využít příkaz:

sudo nginx -t

Tento příkaz vypíše, zdali je možné aktuální konfiguraci použít a jestli ne, tak vypíše chybu, podle které je možno odhalit závadu v konfiguraci.

Pro zavedení konfigurace je možné využít příkaz

sudo nginx -s reload

Tento příkaz konfiguraci zavede bez resetování celé aplikace.

## Databázový server MariaDB

### Konfigurace MariaDB

Konfigurační soubory pro MariaDB se nachazí na cestě /etc/mysql/. V tomto adresáři se nachází soubor námi upravovaná konfigurace v souboru my.cnf. Tento soubor má také INI strukturu, ale mezi názvem proměnné a její hodnotou je znak „rovná se“ (=). Do tohoto souboru byla zkopírována výchozí konfigurace ze stránky [https://docs.nextcloud.com/](https://docs.nextcloud.com/server/latest/admin_manual/configuration_database/linux_database_configuration.html). Jelikož nemám osobní znalosti s konfigurací databází, tak nedokáži většinu nastavení vysvětlit.

Důležitá nastavení pro naše účely ovšem jsou:

* „socket = /run/mysqld/mysqld.sock“ – Pomocí tohoto soketu budou některé naše aplikace komunikovat s databází. Je dobré znát jeho lokaci
* „character\_set\_server = utf8mb4“ – Výchozí znaková sada bude nastavena na UTF-8 s podporou emotikonů, což zaručuje správné fungování česky zapsaných hodnot i s emotikony.

### Vytvoření nové databáze

Všechny příkazy pracující s databázemi je nutné dělat v prostředí databázového serveru, do které je možné se dostat pomocí příkazu:

sudo mysql

Příkaz pro vytvoření nové databáze má doporučenou následující syntaxi:

CREATE DATABASE IF NOT EXISTS název\_databáze CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci;

Doplněním názvu databáze do příkazu se vytvoří nová databáze se znakovou sadou UTF-8, do které bude muset být udělen přístup některému z uživatelů databáze. Je nutné si uvědomit, že uživatelé databáze se vytváří v databázové aplikaci a nemají spojitost s uživateli s linuxovém prostředí.

### Vytvoření nového uživatele databáze

Pro vytvoření nového uživatele v databázovém serveru je doporučeno použít následující příkaz:

CREATE USER 'jméno\_uživatele'@'localhost' IDENTIFIED BY 'heslo';

Po doplnění jména uživatele a bezpečného hesla se vytvoří nový uživatel pomocí kterého se bude možné přihlásit do databázového serveru pomocí příkazu:

sudo mysql -u jméno\_uživatele -p

Poté se bude po uživateli vyžadovat zadání hesla a bude mu udělen přístup k přiděleným databázím přes prostředí databázového serveru.

### Přidání práv pro správu databáze uživateli

Pomocí následujícího příkazu se přidělí specifikovanému uživateli práva pro veškerou správu specifikované databáze a tabulek v ní.

GRANT ALL PRIVILEGES on název\_databáze.\* to 'jméno\_uživatele'@'localhost';

Pro odebrání těchto práv je možné využít následující příkaz:

REVOKE ALL PRIVILEGES on název\_databáze.\* to 'jméno\_uživatele'@'localhost';

### Zápis změn v právech uživatelů

Po úpravě práv uživatelů k databázím je možné použít následující příkaz, který zapíše tyto změny. I když je tento příkaz dobrovolný, je doporučeno ho použít:

FLUSH privileges;

Až poté by se mělo z databázového serveru odcházet pomocí příkazu:

quit;

### Proces MariaDB

Jelikož program MariaDB nahrazuje program MySQL jsou veškeré názvy cest a proměnných tohoto balíčku ve skutečnosti „mysql“. Proces MariaDB se ovládá pomocí příkazů pro procesy systemd. Název procesu je „mysql.service“. Pro syntaxi příkazů viz [Démon systemd](#_Démon_systemd).

## PHP

### Konfigurace PHP

Popis změn konfigurace PHP byl uveden v dokumentaci realizace tohoto počítače v sekci „Nextcloud“. Konfigurační soubory serveru PHP jsou velmi obsáhlé, ovšem každá možnost u sebe má dokumentaci její funkce. Pro naše účely byly upraveny dva soubory.

Souboru na cestě /etc/php/8.0/fpm/php.ini byla na řádku 430 přepsána hodnota pro maximální využití operační paměti na hodnotu „1G“, která znamená 1 GB.

Dále byly v souboru /etc/php/8.0/fpm/pool.d/www.conf nastaveny cesty pro proměnné PATH, TMP, TMPDIR a TEMP. Proměnná PATH určuje, k jakým programům má PHP přístup pro využití pro své procesy. Proměnné TMP, TMPDIR a TEMP poskytují cestu pro dočasné úložiště pro soubory těchto procesů. Dále zde byla nastavená proměnná HOSTNAME, která určuje jméno počítače, na kterém PHP běží.

Složka konfiguračních souborů pro PHP serveru na nachází na cestě /etc/php/8.0/fpm. V této složce je možné najít spoustu obsáhlých konfiguračních souborů, ve kterých je vysvětlena funkce každého možného nastavení. Osobně jsem se moc o tyto nastavení nezajímal a dodržoval jsem akorát nastavení potřebné pro chod použitých aplikací.

### Proces php-fpm

Proces našeho PHP serveru se také ovládá pomocí příkazů pro procesy systemd. Název procesu je „php8.0-fpm.service“. Pro syntaxi příkazů viz [Démon systemd](#_Démon_systemd).

## Cockpit

Tento balíček obsahuje celkem jednoduchou konfiguraci v podobě INI formátovaného souboru na cestě /etc/cockpit/cockpit.conf. V této konfiguraci byli definovány pouze tři nastavení ve skupině „WebService“.

[WebService]

Origins = https://jeff.sspbrno.cz wss://jeff.sspbrno.cz

ProtocolHeader = X-Forwarded-Proto

UrlRoot = /admin

První nastavení „Origin“ definovalo z jaké domény mohou být stránky poskytované. Jestliže by nastal pokus o načtení stránky z jiné domény, tak aplikace neposkytne své služby. Toto nastavení je důležité pro bezpečnost.

Druhé nastavení „ProtocolHeader“ určuje z jaké hlavičky z NGINX má aplikace vyčíst používaný protokol pro komunikaci. Pro definici hlavičky viz [konfigurace NGINX](#_Webový_s_reverzní).

Poslední nastavení „UrlRoot“ určuje pod jakým adresářem bude Cockpit přístupný z uvedené domény. V našem případě se jednalo o adresář „/admin“.

Konfigurační soubor Cockpit obsahuje více možností – viz oficiální dokumentace.

## Nextcloud

### Konfigurační soubor Nextcloud

Konfigurační soubor pro aplikaci nextcloud se nachází na adrese /var/www/nextcloud/config/config.php. Tento soubor byl vytvořen instalačním procesem a neměl by být šířen, jelikož obsahuje přístupové údaje.

V této konfiguraci je spousta zajímavých nastavení, které jsou níže vysvětleny:

* List „trusted\_domains“ – Nextcloud zpracovává pomocí které domény je klient připojen k této aplikaci. Každá adresa v tomto listu je právoplatná pro přístup, jakákoli jiná bude při pokusu o připojení odmítnuta ve webovém rozhraní.
* „default\_phone\_region“ – Výchozí předvolba pro telefonní čísla. Bylo nutné nastavit pro odstranění upozornění o její absenci
* „dbname“, „dbhost“, „dhport“, „dbtableprefix“, „mysql.trf8mb4“, „dbuser“, „dbpassword“ – Obsahují konfiguraci pro komunikaci s databází
* „memcache.local“ – Nastavení určují modul pro použití paměti cache

### Proces Nextcloud

Nextcloud, jelikož se jedná o PHP aplikaci, nemá proces, který by na pozadí zpracovával data. Veškeré práce s daty se provádí při žádostech na PHP souborech pomocí PHP serveru. Jestliže nikdo tyto PHP soubory nežádá, žádné data se nezpracovávají. Toto samozřejmě způsobuje problémy s časově orientovanými úkoly.

### Pravidelná žádost pro aktualizaci dat

Z tohoto důvodu byl do systemd zaveden časovač a proces se jménem „nextcloudcron.service“ a „nextcloudcron.timer“. Tento časovač je nakonfigurovaný následovně:

[Unit]

Description=Run Nextcloud cron.php every 5 minutes

[Timer]

OnBootSec=5min

OnUnitActiveSec=5min

Unit=nextcloudcron.service

[Install]

WantedBy=timers.target

Jedná se o konfiguraci vysvětlenou v sekci [Nastavení časovačů pro systemd](#_Nastavení_časovačů_pro). Tento časovač spouští proces, který vypadá následovně:

[Unit]

Description=Nextcloud cron.php job

[Service]

User=www-data

ExecStart=php8.0 --define apc.enable\_cli=1 -f /var/www/nextcloud/cron.php

KillMode=process

Tento proces má jediný úkol, a to spustit pomocí programu php8.0 soubor /var/www/nextcloud/cron.php, který aktualizuje data této aplikace.

Pro syntaxi příkazů pro práci s časovačem viz [Démon systemd](#_Démon_systemd).

## PCMS

### Databáze

Pomocí informací v sekci [Databázový server MariaDB](#_Databázový_server_MariaDB) je možné vytvořit a pracovat s databází pro PCMS.

### Konfigurační soubor

V kořenovém adresáři aplikace byl vytvořen soubor .env, který určuje způsob připojení k databázi, URL adresy pod kterými budou jednotlivé stránky fungovat a jejich zabezpečení pomocí http hlaviček. U každé z nastavených hodnot se nachází vysvětlení její funkce pomocí poznámky

### Proces systemd

Pro obě aplikace byl vytvořen proces systemd v adresáři /etc/systemd/system pojmenované „pcms-backend.service“ a „pcms-frontend.service“. Obsah procesu pro backend:

[Unit]

Description=PCMS API & Admin service

After=network.target

[Service]

Type=simple

User=www-data

Group=www-data

Environment=NODE\_ENV=production

WorkingDirectory=/srv/sspbrnoweb-monorepo

ExecStart=/usr/bin/env pnpm start:backend

Restart=always

[Install]

WantedBy=multi-user.target

V tomto procesu je důležité zachovat uživatele www-data pro správný přístup k datům a poskytnout proměnou „NODE\_ENV“ v hodnotě „production“ pro rychlejší běh aplikace.

Obsah procesu pro frontend:

[Unit]

Description=PCMS frontend service

[Service]

Type=simple

User=www-data

Group=www-data

Environment=PORT=6530

Environment=NODE\_ENV=production

WorkingDirectory=/srv/sspbrnoweb-monorepo

ExecStart=/usr/bin/env pnpm start:frontend

Restart=always

[Install]

WantedBy=multi-user.target

V tomto procesu bylo opět nutné zachovat uživatele a poskytnou proměnné „PORT“ a „NODE\_ENV“. Zde poskytujeme proměnnou „PORT“ pro správnou komunikaci mezi procesy pomocí TCP.

Oba tyto soubory byly poskytnuty jako příloha. Pro další informace doporučuji navštívit buďto dokumentaci na stránkách [https://bitbucket.org/](https://bitbucket.org/purkynka-renewal/sspbrnoweb-monorepo/src/master/) či maturitní práci pana Adama Žingora.

# Oficiální dokumentace balíčků

|  |  |
| --- | --- |
| Apt | [https://wiki.debian.org/](https://wiki.debian.org/PackageManagement) |
| Systemd | <https://www.freedesktop.org/>  Neoficiální: [https://www.digitalocean.com/](https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files) |
| UFW | [http://manpages.ubuntu.com/](http://manpages.ubuntu.com/manpages/jammy/en/man8/ufw.8.html)  Neoficiální: [https://www.digitalocean.com/](https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-with-ufw-on-ubuntu-18-04) |
| Fail2ban | [https://www.fail2ban.org/](https://www.fail2ban.org/wiki/index.php/Main_Page) |
| SSH server | [https://linux.die.net/](https://linux.die.net/man/5/sshd_config) |
| NGINX | [https://www.nginx.com/](https://www.nginx.com/resources/wiki/) |
| MariaDB | [https://mariadb.com/](https://mariadb.com/kb/en/configuring-mariadb-with-option-files/) |
| PHP | [https://www.php.net/](https://www.php.net/manual/en/configuration.file.php) |
| Cockpit | [https://cockpit-project.org/](https://cockpit-project.org/guide/latest/cockpit.conf.5) |
| Nextcloud | [https://docs.nextcloud.com/](https://docs.nextcloud.com/server/latest/admin_manual/installation/) |
| PCMS | [https://bitbucket.org/](https://bitbucket.org/purkynka-renewal/sspbrnoweb-monorepo/src/master/) |