1. Podizanje HyperV

https://ubuntu.com/download/desktopVirtual switch manager – definirti switch

Prikazivanje IP adrese

**ip addr show**

ip a

**Promjena root PWD:**

sudo passwd root

**Promjena vlasništva na direktoriju**

sudo chown -R $USER:$USER /var/www/html/ostechnix1.lan/public\_html

**Install apache on Linux**

Sudo apt update

Sudo apt install apache2

**Konfiguracija Appache**

[**https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-22-04**](https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-22-04)

$ sudo nano /etc/apache2/sites-enabled/000-default.conf

Restart Appache:

sudo systemctl restart apache2

**Mjesta gdje možeš pronaći .htaccess na Apache2:**

1. **Root direktorij web stranice:**
   * .htaccess datoteka se najčešće nalazi u root direktoriju vaše web stranice. Ovisno o konfiguraciji vašeg servera, to će biti jedan od sljedećih direktorija:
     + /var/www/html/ (Ako koristite osnovnu instalaciju)
     + /var/www/yourdomain.com/ (Ako imate nekoliko web stranica)
     + /home/username/public\_html/ (Ako koristite shared hosting)

Na primjer, ako vaša stranica koristi direktorij /var/www/html/, .htaccess bi se trebala nalaziti tamo.

1. **Konfiguracija MySql**
   * sudo apt install mysql-server
2. sudo mysql
3. ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'password';
4. Mysql exit;
   * sudo mysql\_secure\_installation (promjena pwd za root)
   * mysql -u root -p (pristup bazi s root korisnikom)
5. Instalacija PHP
   * sudo apt install php libapache2-mod-php php-mysql
   * php -v
   * PHP ini:
   * /etc/php/7.4/apache2/php.ini
6. **Composer:**
   * **Korak 1: Preuzmi Composer installer**

[**https://www.digitalocean.com/community/tutorials/how-to-install-and-use-composer-on-ubuntu-18-04**](https://www.digitalocean.com/community/tutorials/how-to-install-and-use-composer-on-ubuntu-18-04)

cd ~

php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"

* + **Korak 2: Verificiraj hash**

cd ~

curl -sS [https://getcomposer.org/installer -o /tmp/composer-setup.php](https://getcomposer.org/installer%20-o%20/tmp/composer-setup.php)

HASH=`curl -sS [https://composer.github.io/installer.sig`](https://composer.github.io/installer.sig%60)

echo $HASH

php -r "if (hash\_file('SHA384', '/tmp/composer-setup.php') === '$HASH') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP\_EOL;"

* + **Instalacja composer-a**

sudo php /tmp/composer-setup.php --install-dir=/usr/local/bin --filename=composer

PRIMJER:

1. **Kreiranje mape za projekat**

Sada, kreiraj traženu mapu **ComposerExercise** u direktoriju /var/www/html/BackendDeveloper.

**Korak 1**:Promjeni vlasnika na BackendDeveloper

sudo chown -R $USER:$USER /var/www/html/BackendDeveloper

**Korak 2: Kreiraj mapu**

sudo mkdir -p /var/www/html/BackendDeveloper/ComposerExercise

**Korak 2: Prebaci se u novokreiranu mapu**

cd /var/www/html/BackendDeveloper/ComposerExercise

**3. Instaliraj paket sa Packagist-a**

Za primjer, instalirat ćemo paket guzzlehttp/guzzle (popularni HTTP client paket).

**Korak 1: Pokreni Composer inicijalizaciju u mapi**

composer init

Ovdje ćeš unijeti osnovne informacije o projektu (ime, opis itd.).

**Korak 2: Instaliraj željeni paket**

composer require guzzlehttp/guzzle

**4. Provjeri instalaciju**

Nakon instalacije paketa, sve zavisne datoteke bit će preuzete u mapu vendor/, a composer.json i composer.lock datoteke će biti kreirane. Provjeri sadržaj mape:

**ls -l**

* strukturu projekta s composer.json, composer.lock i vendor/ mapa.

1. **Git:**

* kreirati direktorij
* pokrenuti git init – definira se glavna grana
* preimenovati master u main – **git branch -M main**
* dodati neku datoteku u master granu
* pokrenuti git add .
* pokrenuti git commit -m „App init“
* git branch – pokazuje grane

Na main branchu se drži kod koji je u produkciji, na njemu se ne radi

Git branch naredba pokazuje u kojem smo branchu – pokazano sa **\***

Prebacivanje na drugu granu

Git checkout „ime grane“

Stvaranje nove grane i automatski prebacivanje

Git checkout -b „naziv grane“

Nazivi za grane: <https://gist.github.com/qoomon/5dfcdf8eec66a051ecd85625518cfd13>

**Brisanje brancheva**

* Pozicioniranje u branch koji nećemo brisati
* Moramo se pozicionirati u main - **git checkout main**
* Brisanje - git branch -d „ime brancha“
* Ako ima neprenesenih promjena force delete **git branch -D „ime brancha“**

**Spajanje promjena n jednom branchu u drugi**

* Pozicionirani u branch u koji ćemo mergati
* **Git merge featureA**

**-prebaci i commit**

**Ako radimo na istoj datoteci u dvije različite grane, a želimo promjene comitati u treću**

1. **Pokrenemo git add .**
2. **Pokrenemo git commit -m „opis priomjene“ na svakoj grani**
3. **Prebacimo se u granu u kojoj želimo spojiti promjene:**
4. **Pokrenemo git merge „naziv grane 1“**
5. **Pokrenemo git merge „naziv grane 2“**

**U VS studio odaberemo što želimo napraviti s promjenama i nakon toga pokrenemo**

**Git add . i git commit – m „Fix ….“**

**Git log - popis svih commiteva**

**git checkout <commit-hash> - vraćanje na neki od commiteva**

**Udaljeni repozitorij:**

Github /dashboard

Prijenos https protokol – uvijek pita za lozinku kad želimo nešto prenijeti na repozitoriji, traži svaki puta prijavu

Ssh protokol – uspostavlja sigurni vezu i mehanizam s kojim osiguravamo da se naše računalo predstavi git hubu bez unošenja lozinke

**Ssh ključevi**

* Prenese javni ključ u git hub servis – predstavljamo se s tim ključem

[**https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent**](https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent)

**Pokrenemo:**

**ssh-keygen -t ed25519 -C "your\_email@example.com"**

**odaberemo naziv ključa**

**generira dvije datoteke : javnu i privatnu (javni i privatni ključ)ž**

**javni ključ se daje servisima i na taj način komuniciramo, privatni ne ide nigdje**

**kopiramo public ključ u notepadu i odemo na git hub**

**Slika na kojoj se prikazuje tekst, softver, snimka zaslona, broj

Opis je automatski generiran**

**Kreirati udaljeni repozitorij**

Gumb novi repozitorij and

Prebaciti s https u ssh

git remote add origin git@github.com:vesnav111/git-branches.git

git remote -v – provjera statusa

git push -u origin „naziv grane“

git branch -a - pokazuje sve grane i online

Povezivanje s dev grane na main granu online

Zadnji korak odabrati „Megre pull request“ - odabrati „rebase and merge“ – da se branchevi svedu na istu razinu

Pull request – smisao ima jedino kad se radi u timu, šalje se na odobrenje, team lead prihvaća je li kod u redu

**Semantički sustav verzioniziranja:**

Semantički sustav verzioniranja (eng. Semantic Versioning, često skraćeno kao SemVer) je standard za označavanje verzija softvera koji pomaže programerima i korisnicima razumjeti promjene u softveru na temelju brojeva verzija. Koristi format MAJOR.MINOR.PATCH (npr. 1.4.2), gdje svaki broj ima jasno definirano značenje:

Komponente verzije:

MAJOR (glavni broj):

Mijenja se kada se uvedu promjene koje nisu kompatibilne s prethodnim verzijama (tzv. "breaking changes").

Npr. prelazak s 1.x.x na 2.0.0 označava da su određene funkcionalnosti promijenjene ili uklonjene te možda neće raditi s kodom koji koristi stariju verziju.

MINOR (manji broj):

Mijenja se kada se dodaju nove funkcionalnosti koje su kompatibilne s postojećom verzijom.

Npr. prelazak s 1.3.0 na 1.4.0 označava dodavanje novih značajki bez utjecaja na postojeći kod.

PATCH (zakrpa):

Mijenja se kada se isprave greške ili uvedu manje izmjene koje ne mijenjaju funkcionalnost (kompatibilne su).

Npr. prelazak s 1.3.2 na 1.3.3 označava ispravak buga bez utjecaja na ponašanje sustava.

Pravila semantičkog verzioniranja:

Verzija započinje s 0.1.0 tijekom razvoja (prije stabilnog izdanja). Verzija 0.x.y se smatra nestabilnom i može se često mijenjati.

Kada softver postane stabilan, prelazi na 1.0.0.

Brojevi se povećavaju na temelju vrsta promjena:

"Breaking changes" -> povećava MAJOR.

Dodavanje kompatibilnih značajki -> povećava MINOR.

Ispravci grešaka -> povećava PATCH.

**KLONIRANJE REPOZITORIJA**

* **Kreira se repozitorij**
* **Nakon toga odabere se suradnik**
* **Suradnik dobije poveznicu**
* **Kopira kod – lokalno računalo + ssh**
* **Lokalno se s git bash pozicioniramo gdje želimo i napravimo clone**
* **Git clone + kopiramo kod**
* **Kreiramo svoju granu**
* **Kad smo gotovi napravimo**
* **Git push -u origin „naziv grane“**

**Dolazi na odobrenje team leadu**

**### Knjiga ###**

**https://tkrajina.github.io/uvod-u-git/git.pdf**

**### Dokumentacija ###**

**https://git-scm.com/docs**

**### Kako generairati SSH ključ ###**

[**https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent**](https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent)