# Next.js

## next-intl

### use for async server component

use getTranslations instead of useTranslations

## Force to compile mode (static or dynamic)

export const dynamic = 'force-static'

export const dynamic = 'force-dynamic'

## next-auth

### GitHubProvider

callback=> /api/auth/callback

### Google

callback => /api/auth/callback/google

# React

## TypeScript

### MouseEvent Type

import type {MouseEvent} from "react";

## Private Route with Context

### lib/auth-utils.tsx

import React, {createContext, useContext, useState} from "react";

interface AuthContextProps {

isAuthenticated: boolean;

loginUser: () => void;

logoutUser: () => void;

}

const AuthContext = createContext<AuthContextProps | undefined>(undefined);

export const AuthProvider = (

{

children,

} : {

children: React.ReactNode,

}

) => {

const [isAuthenticated, setIsAuthenticated] = useState(false);

const loginUser = () => setIsAuthenticated(true);

const logoutUser = () => setIsAuthenticated(false);

return (

<AuthContext.Provider value={{isAuthenticated, loginUser, logoutUser}}>

{children}

</AuthContext.Provider>

)

}

export const useAuth = (): AuthContextProps => {

const context = useContext(AuthContext);

if (!context) {

throw new Error("useAuth muse be used within an AuthProvider");

}

return context;

}

### components/PrivateRoute.tsx

import {Outlet, Navigate, type Path, useLocation} from "react-router-dom";

import {encode} from "js-base64";

import {useAuth} from "@/lib/auth-util";

export default function PrivateRoute(

{

loginUrl,

} : {

loginUrl: string | Partial<Path>

}

) {

const {isAuthenticated} = useAuth();

const {pathname} = useLocation();

return isAuthenticated ? <Outlet /> : <Navigate to={`${loginUrl}?redirect=${encode(pathname)}`} replace />;

}

### pages/LoginPage.tsx

import React, {useState} from "react";

import {useAuth} from "@/lib/auth-util";

const initData = {

email: "admin@example.com",

password: "password",

};

export default function LoginPage() {

const {loginUser} = useAuth();

const [email, setEmail] = useState(initData.email);

const [password, setPassword] = useState(initData.password);

const login = (e: MouseEvent) => {

e.preventDefault();

if (email === initData.email && password === initData.password) {

console.log("loged in")

loginUser();

}

}

//….

}

### App.tsx

import {BrowserRouter as Router, Routes, Route} from "react-router-dom";

import LoginPage from "@/pages/LoginPage.tsx";

import HomePage from "@/pages/HomePage.tsx";

import AccountPage from "@/pages/AccountPage.tsx";

import Navbar from "@/components/Navbar";

import PrivateRoute from "@/components/PrivateRoute.tsx";

import {AuthProvider} from "@/lib/auth-util.tsx";

import "./App.css";

function App() {

return (

<AuthProvider>

<Router>

<Navbar />

<Routes>

<Route path="/login" element={<LoginPage />} />

<Route path="/" element={<HomePage />} />

<Route path="/" element={<PrivateRoute loginUrl={"/login"}/>} >

<Route path="/account" element={<AccountPage />} />

</Route>

</Routes>

</Router>

</AuthProvider>

)

}

export default App

### Navbar.tsx

import {Link, useNavigate} from "react-router-dom";

import {Disclosure} from "@headlessui/react";

import {ArrowLeftEndOnRectangleIcon, ArrowLeftStartOnRectangleIcon } from "@heroicons/react/24/outline";

import {useAuth} from "@/lib/auth-util";

const navigation = [

{ name: "Home", href: "/" },

{ name: "Account", href: "/account" },

];

export default function Example() {

const {isAuthenticated, logoutUser} = useAuth();

const navigate = useNavigate();

return (

<Disclosure as="nav" className="bg-gray-800">

<div className="mx-auto max-w-7xl px-2 sm:px-6 lg:px-8">

<div className="relative flex h-16 items-center justify-between">

<div className="flex flex-1 items-center justify-center sm:items-stretch sm:justify-start">

<div className="hidden sm:ml-6 sm:block">

<div className="flex space-x-4">

{navigation.map((item) => (

<Link

key={item.name}

to={item.href}

className="text-gray-300 hover:bg-gray-700 hover:text-white rounded-md px-3 py-2 text-sm font-medium"

>

{item.name}

</Link>

))}

</div>

</div>

</div>

<div className="absolute inset-y-0 right-0 flex items-center pr-2 sm:static sm:inset-auto sm:ml-6 sm:pr-0">

<button

type="button"

className="relative rounded-full bg-gray-800 p-1 text-gray-400 hover:text-white focus:outline-none focus:ring-2 focus:ring-white focus:ring-offset-2 focus:ring-offset-gray-800" onClick={() => isAuthenticated ? logoutUser() : navigate("/login")} >

<span className="absolute -inset-1.5" />

<span className="sr-only">View notifications</span>

{isAuthenticated && <ArrowLeftStartOnRectangleIcon className="h-6 w-6" aria-hidden="true" />}

{!isAuthenticated && <ArrowLeftEndOnRectangleIcon className="h-6 w-6" aria-hidden="true" />}

</button>

</div>

</div>

</div>

</Disclosure>

)

}

## Using localStorage by custom hook

### useLocalStorage.js

import { useState, useEffect } from "react";

function getStorageValue(key, defaultValue) {

// getting stored value

const saved = localStorage.getItem(key);

const initial = JSON.parse(saved);

return initial || defaultValue;

}

export const useLocalStorage = (key, defaultValue) => {

const [value, setValue] = useState(() => {

return getStorageValue(key, defaultValue);

});

useEffect(() => {

// storing input name

localStorage.setItem(key, JSON.stringify(value));

}, [key, value]);

return [value, setValue];

};

### components/Form1.jsx

import { useLocalStorage } from "../useLocalStorage";

const Form1 = () => {

const [name, setName] = useLocalStorage("name", "");

return (

<form>

{/\* ... \*/}

</form>

);

};

export default Form1;

### SSR Rendering like Next.js

function getStorageValue(key, defaultValue) {

// getting stored value

if (typeof window !== "undefined") {

const saved = localStorage.getItem(key);

const initial = saved !== null ? JSON.parse(saved) : defaultValue;

return initial;

}

}

# Prisma

## Init

npx prisma init

## Synchronize schema to DB

npx prisma db pull

## Where for joined table's field

prisma.invoices.findMany({

include: {

customer: true,

},

where: {

OR: [

{

customer: {

name: {

contains: query,

mode: "insensitive",

},

},

}, {

customer: {

email: {

contains: query,

mode: "insensitive",

},

},

}, {

status: {

contains: query,

mode: "insensitive",

},

},

// amount: {

// contains: query,

//

// }

// date: {

//

// },

],

}

})

# TypeScript

## type vs interface

### Common

Both can define a data type.

Types aliases in TypeScript mean "a new for anytype".

An interface defines a contract that an object must adhere to.

### Difference

#### － Primitive types

We can use define a type alias for a primitive type as below:

type Alias = string;

type NullOrUndefined = null | undefined;

But, we can't use an interface to alias a primitive type.

#### － Union types

Union types allow us to describe values that can be one of several types and create unions of various primitive, literal, or complex types:

type Transport = 'Bus' | 'Car' | 'Bike' | 'Walk';

Union type can only be defined using type. There is no equivalent to a union type in an interface. But, it is possible to create a new union type from two interfaces, like so:

interface CarBattery {

power: number;

}

interface Engine {

type: string;

}

type HybridCar = Engine | CarBattery;

#### － Function types

In typescript, a function type represents a function's type signature. Using the type alias, we need to specify the parameters and the return type to define a function type:

type AddFn = (num1: number, num2: number) => number;

We can also use an interface to represent the function type:

interface IAdd {

(num: number, num2: number): number;

}

Both type and interface similarly define function types, except for a subtle syntax difference of interface using: vs. => when using type. Type is preferred in this case because it's shorter and thus easier to read.

Another reason to use type for defining a function type is its capabilities that the interface lacks. Here's an example:

type Car = 'ICE' | 'EV';

type ChargeEV = (kws: number)=> void;

type FillPetrol = (type: string, liters: number) => void;

type RefillHandler<A extends Car> = A extends 'ICE' ? FillPetrol : A extends 'EV' ? ChargeEV : never;

const chargeTesla: RefillHandler<'EV'> = (power) => {

// Implementation for charging electric cars (EV)

};

const refillToyota: RefillHandler<'ICE'> = (fuelType, amount) => {

// Implementation for refilling internal combustion engine cars (ICE)

};

#### － Declaration merging

Declaration merging is a feature that is exclusive to interfaces

interface Client {

name: string;

}

interface Client {

age: number;

}

const harry: Client = {

name: 'Harry',

age: 20,

};

When used in the right places, declaration merging can be very usefule. One common use case for declaration merging is to extend a 3rd-party's library's type definition to fit the needs of a particular project.

#### － Extends vs. intersection

An interface can extend one or multiple interfaces.

For example, we can create a VIPClient interface by extending the Client interface.

interface VIPClient extends Client {

benefits: string[];

}

To achieve a similar result of types, we need to use an intersection operator:

type VIPClient = Client & {

benefits: string[];

}

You can also extends an interface from a type alias with statically known members:

type Client = {

name: string;

}

interface VIPClient extends Client {

benefits: string[];

}

The exception is union types. If you try to extend an interface from a union type, you'll receive the following error:

type Jobs = 'salary worker' | 'retired';

interface MoreJobs extends Jobs {

description: string;

}

Type aliases can extend interfaces using the intersection, as below:

interface Client {

name: string;

}

Type VIPClient = Client & { benefits: string[]};

#### － Handling conflicts when extending

When extending interfaces, the same property key isn't allowed, as in the example below:

interface Person {

getPermission: () => string;

}

interface Staff extends Person {

getPermission: () => string[];

}

An error is thrown because a conflict is detected.

Type aliases handle conflicts differently. In the case of a type alias extending another type with the same property key, it will automatically merge all properties instead of throwing errors.

type Person = {

getPermission: (id: string) => string;

};

type Staff = Person & {

getPermission: (id: string[]) => string[];

};

const AdminStaff: Staff = {

getPermission: (id: string | string[]) =>{

return (typeof id === 'string'? 'admin' : ['admin']) as string[] & string;

}

}

It is important to note that the type intersection of two properties may produce unexpected results. In the example below, the name property for the extended type Staff becomes never, since it can't be both string and number at the same time.

type Person = {

name: string

};

type Staff = person & {

name: number

};

// error: Type 'string' is not assignable to type 'never'.(2322)

const Harry: Staff = { name: 'Harry' };

#### － Implementing classes using interfaces or type aliases

In TypeScript, we can implement a class using either an interface or a type alias:

interface Person {

name: string;

greet(): void;

}

class Student implements Person {

name: string;

greet() {

console.log('hello');

}

}

type Pet = {

name: string;

run(): void;

};

class Cat implements Pet {

name: string;

run() {

console.log('run');

}

}

As shown above, both interfaces and type aliases can be used to implement a class similarly; the only difference is that we can't implement a union type.

type primaryKey = { key: number; } | { key: string; };

// can not implement a union type

class RealKey implements primaryKey {

key = 1

}

#### － Working with tuple types

In Typescript, the tuple type allows us to express an array with a fixed number of elements, where each element has its data type. It can be useful when you need to work with arrays of data with a fixed structure:

type TeamMember = [name: string, role: string, age: number];

Interface don't have direct support for tuple types. Although we can create workarounds like in the example below, it is not as concise or readable as using the tuple type:

interface ITeamMember extends Array<string | number>

{

0: string; 1: string; 2: number

}

const peter: ITeamMember = ['Harry', 'Dev', 24];

#### － When to use types vs. interfaces

In many cases, they can be used interchangeably depending on personal preference. But, we should use type aliases in the following use cases:

－To create a new name for a primitive type

－To define a union type, tuple type, function type, or another more complex type

－To overload functions

－To use mapped types, conditional types, type guards, or other advanced type features

Below is an example of the advanced type feature that the interface can't achieve.

type Client = {

name: string;

address: string;

}

type Getters<T> = {

[K in keyof T as `get${Capitalize<string & K>}`]: () => T[K];

};

type ClientType = Getters<Client>;

// type clientType = {

// getName: () => string;

// getAddress: () => string;

// }

## Compiling using Webpack (Express)

### console

//package.json

"buildProd": "rimraf build/server.production.js && webpack --mode production --config webpack.config.js",

"buildDev": "rimraf build/server.development.js && webpack --mode development --config webpack.config.js --watch"

### Webpack config file

//webpack.config.js

const path = require("path");

const nodeExternals = require("webpack-node-externals");

const { RunScriptWebpackPlugin } = require("run-script-webpack-plugin");

module.exports = (env, argv) => {

const SERVER\_PATH = "./src/server.ts";

return {

entry: {

server: SERVER\_PATH,

},

output: {

path: path.join(\_\_dirname, "backend"),

publicPath: "/",

filename: "[name]." + argv.mode + ".js",

},

mode: argv.mode,

target: "node",

node: {

// Need this when working with express, otherwise the build fails

\_\_dirname: false, // if you don't put this is, \_\_dirname

\_\_filename: false, // and \_\_filename return blank or /

},

devtool: "inline-source-map",

resolve: {

extensions: [".tsx", ".ts", ".js"],

modules: [path.resolve("./src/")],

alias: {

\_app: path.resolve("./src/"),

},

},

externals: [nodeExternals()], // Need this to avoid error when working with Express

module: {

rules: [

{

test: /\.tsx?$/,

use: "ts-loader",

exclude: /node\_modules/,

},

],

},

plugins:

argv.mode === "development"

? [

new RunScriptWebpackPlugin({

name: "server.development.js",

nodeArgs: ["--inspect"],

autoRestart: true,

keyboard: true,

}),

]

: [],

};

};

# TailwindCSS

## Apply to React + Vite

Install Tailwind CSS and Other Dependencies

npm install -D tailwindcss postcss autoprefixer

Generate the Configuration Files

npx tailwindcss init -p

Configure Source Paths

/\*\* @type {import('tailwindcss').Config} \*/

export default {

content: [

"./index.html",

"./src/\*\*/\*.{js,js,jsx,tsx}",

],

theme: {

extend: {},

},

plugins: [],

}

Add Tailwind Directive to Your CSS

@tailwind base;

@tailwind components;

@tailwind utilities;

Enjoy Tailwind CSS

# ASP.NET Core

## Migrations

### Create a migration

dotnet ef migrations add Initial

### Reset the database

dotnet ef database drop --force --context StoreDbContext

### Apply the identity migration

dotnet ef migrations add Initial --context AppIdentityDbContext

dotnet ef database update --context AppIdentityDbContext

# PostgreSQL

## Configure PostgreSQL for Remote Access

By default, PostgreSQL is configured for local access only. If your application and database servers are hosted on a different server, then you must configure PostgreSQL for remote access. You can configure it by editing the file pg\_hba.conf:

nano /etc/postgresql/14/main/pg\_hba.conf

Find the following line:

local all all peer

And replace it with the following line:

local all all trust

Next, add the following line:

host all all 0.0.0.0/0 md5

Save and close the file when you have finished.

Next, you will also need to edit the PostgreSQL main configuration file and change the listening port:

nano /etc/postgresql/14/main/postgresql.conf

Change the following line:

listen\_addresses='\*'

Save and close the file, then restart the PostgreSQL service to apply the changes:

systemctl restart postgresql

You can now check the PostgreSQL listening port using the following command:

ss -antpl | grep 5432

You will get the following output:

LISTEN 0 244 0.0.0.0:5432 0.0.0.0:\* users:(("postgres",pid=19030,fd=5))

LISTEN 0 244 [::]:5432 [::]:\* users:(("postgres",pid=19030,fd=6))

## Create database

create database dbname;

grant all privileges on database dbname to dbuser;

\l

## Execute SQL script from Terminal

psql -h your\_host -U your\_username -d your\_database -f script.sql

## Switch Database and list tables

\c databasename

\dt

## Switch owner of schema

ALTER SCHEMA public OWNER TO your\_user;

# Qt

## QTableView lazy loading

### FileListModel.cpp

#include "filelistmodel.h"

#include <QGuiApplication>

#include <QDir>

#include <QPalette>

#include "qdebug.h"

FileListModel::FileListModel(QObject \*parent)

: QAbstractTableModel(parent), fileCount(0) //edit

{}

int FileListModel::rowCount(const QModelIndex &parent) const

{

return parent.isValid() ? 0 : fileCount;

}

QVariant FileListModel::data(const QModelIndex &index, int role) const

{

if (!index.isValid())

{

return QVariant();

}

if (index.row() >= fileList.size() || index.row() < 0)

{

return QVariant();

}

if (role == Qt::DisplayRole)

{

return fileList.at(index.row());

}

return QVariant();

}

bool FileListModel::canFetchMore(const QModelIndex &parent) const

{

if (parent.isValid())

{

return false;

}

return (fileCount < fileList.size());

}

int FileListModel::columnCount(const QModelIndex &parent) const

{

return parent.isValid() ? 0 : colCount;

}

void FileListModel::fetchMore(const QModelIndex &parent)

{

if (parent.isValid())

{

return;

}

int remainder = fileList.size() - fileCount;

int itemsToFetch = qMin(100, remainder);

if (itemsToFetch <= 0)

{

return;

}

beginInsertRows(QModelIndex(), fileCount, fileCount + itemsToFetch - 1);

qDebug()<< "Qmodelindex "<< QModelIndex()<< "filecount "<< fileCount <<"filecount + itemtofetch "<<fileCount + itemsToFetch - 1;

fileCount += itemsToFetch;

endInsertRows();

}

void FileListModel::setColumnNumber(const int x) //edit

{

colCount = x;

}

void FileListModel::setDataToList(const QList<float> &data)

{

beginResetModel();

fileList = data;

fileCount = 0;

endResetModel();

}

### FileListModel.h

#ifndef FILELISTMODEL\_H

#define FILELISTMODEL\_H

#include <QAbstractTableModel>

#include <QStringList>

class FileListModel : public QAbstractTableModel //edit

{

Q\_OBJECT

public:

FileListModel(QObject \*parent = nullptr);

int rowCount(const QModelIndex &parent = QModelIndex()) const override;

QVariant data(const QModelIndex &index, int role = Qt::DisplayRole) const override;

int columnCount(const QModelIndex &parent = QModelIndex()) const override; //edit

void setColumnNumber(const int );

public slots:

void setDataToList(const QList<float>&);

protected:

bool canFetchMore(const QModelIndex &parent) const override;

void fetchMore(const QModelIndex &parent) override;

private:

QList<float> fileList;

int fileCount;

int colCount;//edit

};

#endif // FILELISTMODEL\_H

# Git

## Setting user/email

git config --global user.name "FIRST\_NAME LAST\_NAME"

git config --global user.email "MY\_NAME@example.com"

git config --local user.name "FIRST\_NAME LAST\_NAME"

git config --local user.email "MY\_NAME@example.com"

git config --global --unset user.name

git config --global --unset user.email

## Disable auto CRLF

git config --global core.autocrlf false

## Setup for Ubuntu

git config --global user.name 'your username here'

git config --global user.email 'your email here'

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

# ssh-keygen -t rsa -C "your\_email@example.com"

cat ~/.ssh/id\_ed25519.pub

# On Windows

# type C:\Users\Administrator\.ssh\id\_ed25519.pub

Add key into Github settings > SSH and GPG Keys

ssh -T git@github.com

## Clone local repo with branches

git clone source target

## Push to online

git remote add origin https://bitbucket.org/<username>/<repository-name>.git

git push -u origin master

## Change upstream

git branch --set-upstream-to upstream

## Resolve conflict

# switch to current-branch

git pull

git pull origin conflict-branch

# resolve conflict using vscode

git commit

## Rename the branch

git branch -m <old\_name> <new\_name>

## Display history graph

git log --branches --tags --graph --oneline --decorate

## Cancel last commit

git reset --soft HEAD~1

This command undoes the last commit but keeps your changes staged. If you want to remove the commit and also unstage the changes, you can use:

git reset HEAD~1

If you want to undo the commit and discard the changes completely, you can use:

git reset --hard HEAD~1

## Switch to new branch

git checkout -b brachname

## Attach add/remove branch

git branch branchname

git branch -d branchname

## Rebase with Interactive

git rebase --interactive <new base> <branch>

## Fetch all branches

git fetch --all

## Multiple Github account

~/.ssh/config

#Default GitHub

Host github-xx

HostName github.com

User git

IdentityFile ~/.ssh/id\_rsa

## Add/Change origin

git remote add origin <new\_url>

git remote set-url origin <new\_url>

## Stash

### Basic Commands

git stash # Save Changes

git stash list # List Stashes

git stash apply [stash@{index}] # Apply Stash

git stash pop [stash@{index}] # Pop Stash

git stash drop [stash@{index}] # Drop Stash

git stash clear # Clear All Stashes

### Useful Options

git stash -p|--patch # Interactively select parts of the changes to stash

git stash -k|--keep-index # Stash changes but keep the staged changes in the index

git stash -u|--include-untracked # Also stash untracked files

git stash -a|--all # Stash both tracked and untracked files

## Rebase a commit to other base

git rebase --onto <target-base-hash> <current-base-hash> <branch-to-move>

# GitBook

## Publish

You can publish the book when it is finished like this:

Click [Share] button.

Turn on the [Publish this space to the web] switch.

If you want to assign custom domain to that publish, click [Connect a domain] button. Then input a subdomain, then click [Next: Configure DNS] buton. You will get the [Target] domain for published result. You have to connect that with your custom domain. Finally click [Check domain] button. All the things have been done.

# JetBrain

## Beautify by JetBrain

Ctrl + Alt + L

# Serve (npm module)

|  |  |
| --- | --- |
| [public](https://github.com/vercel/serve-handler" \l "public-string) | Set a sub directory to be served |
| [cleanUrls](https://github.com/vercel/serve-handler" \l "cleanurls-booleanarray) | Have the .html extension stripped from paths |
| [rewrites](https://github.com/vercel/serve-handler" \l "rewrites-array) | Rewrite paths to different paths |
| [redirects](https://github.com/vercel/serve-handler" \l "redirects-array) | Forward paths to different paths or external URLs |
| [headers](https://github.com/vercel/serve-handler" \l "headers-array) | Set custom headers for specific paths |
| [directoryListing](https://github.com/vercel/serve-handler" \l "directorylisting-booleanarray) | Disable directory listing or restrict it to certain paths |
| [unlisted](https://github.com/vercel/serve-handler" \l "unlisted-array) | Exclude paths from the directory listing |
| [trailingSlash](https://github.com/vercel/serve-handler" \l "trailingslash-boolean) | Remove or add trailing slashes to all paths |
| [renderSingle](https://github.com/vercel/serve-handler" \l "rendersingle-boolean) | If a directory only contains one file, render it |
| [symlinks](https://github.com/vercel/serve-handler" \l "symlinks-boolean) | Resolve symlinks instead of rendering a 404 error |
| [etag](https://github.com/vercel/serve-handler" \l "etag-boolean) | Calculate a strong ETag response header, instead of Last-Modified |

# AWS

## Stop streaming channel

Go to IVS.

Select [Live Channel] and select the channel to stop.

Then in details page, click [Stop stream]. You can reset channel stream key.

## Create general purpose VPS

Create an instance on Lightsail console

# VSCode

## Open Command Palette

Ctrl+Shift+P

## Format Code

Shift+Alt+F

## Organize Imports

Shift+Alt+O

## Bind new hot key

File>Preferences>Keyboard Shortcuts

## Open file by name

Ctrl+P>Type filename

## Disable Markdown lint

"markdownlint.config": {

"default": false,

}

## Show git log graph

Ctrl+Shift+P>git log>Enter

## Disable reopen last project

Go to File > Preferences > Settings (or Code > Preferences > Settings on macOS)

"window.restoreWindows": "none"

# Windows

## Delete folder from Terminal

rmdir /s /q foldername

## List all items including hidden

rmdir /s /q foldername

## Create Symlink

MKLINK [[/D] | [/H] | [/J]] Link Target

/D Creates a directory symbolic link. Default is a file

symbolic link.

/H Creates a hard link instead of a symbolic link.

/J Creates a Directory Junction.

Link Specifies the new symbolic link name.

Target Specifies the path (relative or absolute) that the new link refers to.

## Check Port

netstat -aon | find "123456"

# Linux

## Nano editor

### Search text

Ctrl+W or F6

## Get PID

pidof chrome

ps ax | grep <name>

## Get IP address by command

ifconfig -a

ip addr (ip a)

hostname -I | awk '{print $1}'

ip route get 1.2.3.4 | awk '{print $7}'

nmcli -p device show

## Check Port

sudo lsof -i -P -n | grep LISTEN

sudo netstat -tulpn | grep LISTEN

sudo ss -tulpn | grep LISTEN

sudo lsof -i:22 ## see a specific port such as 22 ##

sudo nmap -sTU -O IP-address-Here

## Setup SSH Keys on Linux

ssh-keygen -t <dsa|ecdsa|rsa|ed25519> -f <filename> -C "email"

ssh-copy-id username@remote\_host

==============================

# if ssh-copy-id is not exist

cat ~/.ssh/id\_rsa.pub | ssh username@remote\_host "mkdir -p ~/.ssh && touch ~/.ssh/authorized\_keys && chmod -R go= ~/.ssh && cat >> ~/.ssh/authorized\_keys"

# or manually

cat ~/.ssh/id\_rsa.pub # copy this as public\_key\_string

mkdir -p ~/.ssh

echo public\_key\_string >> ~/.ssh/authorized\_keys

chmod -R go= ~/.ssh

chown -R group:user ~/.ssh

==============================

ssh username@remote\_host #using password

sudo nano /etc/ssh/sshd\_config

# /etc/ssh/sshd\_config

PasswordAuthentication no

#

sudo systemctl restart ssh

ssh username@remote\_host @using ssh-key

## Install SSH server

sudo apt update && sudo apt upgrade

sudo apt install openssh-server

sudo systemctl enable --now ssh

sudo systemctl status ssh

sudo ufw status

sudo ufw allow ssh

============================

ssh username@IP\_address

## Setup SSH Tunnel in Background

ssh -L localhost:3306:127.0.0.1:3306 username@host -f

## Install RDP on Ubuntu Server

sudo apt update

sudo apt install xfce4 xfce4-goodies -y

sudo apt install xrdp -y

sudo systemctl status xrdp

sudo systemctl start xrdp

sudo nano /etc/xrdp/xrdp.ini

cd ~

echo "xfce4-session" | tee .xsession

sudo systemctl restart xrdp

## Install VNC on Ubuntu Server

ssh username@your\_vps\_ip

sudo apt update

sudo apt upgrade -y

sudo apt install xfce4 -y

sudo apt install tightvncserver -y

vncserver

vncserver -kill :1

nano ~/.vnc/xstartup

============================

!/bin/bash

xrdb $HOME/.Xresources

startxfce4 &

============================

chmod +x ~/.vnc/xstartup

vncserver

vncserver -geometry 1920x1020

### Install as a Service

sudo nano /etc/systemd/system/vncserver@.service

==========================

[Unit]

Description=Start TightVNC server at startup

After=syslog.target network.target

[Service]

Type=forking

User=ubuntu

Group=ubuntu

WorkingDirectory=/home/ubuntu

PIDFile=/home/ubuntu/.vnc/%H:%i.pid

ExecStartPre=-/usr/bin/vncserver -kill :%i > /dev/null 2>&1

ExecStart=/usr/bin/vncserver -depth 24 -geometry 1920x1020 -localhost :%i

ExecStop=/usr/bin/vncserver -kill :%i

[Install]

WantedBy=multi-user.target

==========================

sudo systemctl daemon-reload

sudo systemctl enable vncserver@1.service

vncserver -kill :1

sudo systemctl start vncserver@1

## Install RealVNC Server

wget https://downloads.realvnc.com/download/file/vnc.files/VNC-Server-7.12.0-Linux-x64.deb

## Install VSCode

sudo snap install code --classic

## Install Google Chrome

wget https://dl.google.com/linux/direct/google-chrome-stable\_current\_amd64.deb

sudo dpkg -i google-chrome-stable\_current\_amd64.deb

## Add Custom Screen Resolution

xrandr -q

cvt 1600 900

sudo xrandr --newmode "1600x900\_60.00" 118.25 1600 1696 1856 2112 900 903 908 934 -hsync +vsync

sudo xrandr --addmode <display> "1600x900\_60.00"

# if you want to add new resolution every start, add commands on profile

nano ~/.profile

# Node.js

## Switch node version when starting terminal

~/.bashrc, ~/.zshrc

# Function to use the Node version specified in .nvmrc

autonvm() {

if [ -f .nvmrc ]; then

nvm use

elif [ -f ../.nvmrc ]; then

(cd .. && autonvm)

elif [ -f ../../.nvmrc ]; then

(cd ../.. && autonvm)

elif [ -f ../../../.nvmrc ]; then

(cd ../../../ && autonvm)

fi

}

# Add this function to the cd command

cd() {

builtin cd "$@" && autonvm

}

# Call autonvm on terminal startup

autonvm

## Yarn install without changing lockfile

yarn install --frozen-lockfile

# Kubernetes

## Install on Ubuntu

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"

echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

##########################################

# If you do not have root access on the target system, you can still

# install kubectl to the ~/.local/bin directory:

chmod +x kubectl

mkdir -p ~/.local/bin

mv ./kubectl ~/.local/bin/kubectl

# and then append (or prepend) ~/.local/bin to $PATH

##########################################

kubectl version --client

kubectl version --client --output=yaml

### Install from k3s.io

curl -sfL https://get.k3s.io | sh -

mkdir -p ~/.kube

sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config

sudo chown $USER:$USER ~/.kube/config

export KUBECONFIG=~/.kube/config # add on ~/.profile or ~/.bashrc

## minicube

### Start a minikube Cluster

minikube start

### Open the Dashboard

minikube dashboard

minikube dashboard --url

### Create Deployment

kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-images/agnhost:2.39 -- /agnhost netexec --http-port=8080

kubectl get deployments

kubectl get pods

kubectl get events

kubectl config view

kubectl logs hello-node-5f76cf6ccf-br9b5

### Create a Service

kubectl expose deployment hello-node --type=LoadBalancer --port=8080

kubectl get services

minikube service hello-node

### Enable Add-ons

Enable addons

minikube addons enable metrics-server

kubectl top pods

minikube addons disable metrics-server

### Clean up

kubectl delete service hello-node

kubectl delete deployment hello-node

minikube stop

minikube delete #optional

### Get Minikube IP & Service Endpoints

minikube ip

minikebe service --url <service-name>

### Create Persistent Volumen on Minikube

# example

apiVersion v1

kind: PersistentVolume

metadata:

name: pv0001

spec:

accessModes:

- ReadWriteOnce

capacity:

storage: 5Gi

hostPath:

path: /data/pv0001/

## Commands

### List nodes

kubectl get nodes

### List Pods

kubectl get pods

kubectl get pods -n demo # specify namespace

kubectl get pods --all-namespaces

### Create a Pod

kubectl run nginx --image nginx:latest

### Create a Deployment

kubectl create deployment nginx --image nginx:latest --replicas 3

kubectl get pods

### Scale a Deployment

kubectl scale deployment nginx --replicas 5

kubectl get pods

### Expose a Service

kubectl expose deployment/nginx --port 80 --type NodePort

kubectl get services

kubectl get nodes -o wide # display internal-ip

### Using Port Forwarding

kubectl delete service nginx

kubectl expose deployment/nginx --port 80

kubectl get services

kubectl port-forward service/nginx 8080:80

kubectl port-forward deployment/nginx 8080:80

## Apply a YAML File

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

containers:

- name: nginx

image: nginx:latest

===========================

kubectl apply -f nginx.yaml

# Prettier and ESLint

## .eslintrc.js | .eslintrc.json

module.exports = {

parser: "@typescript-eslint/parser",

env: {

es6: true,

node: true,

},

extends: ["plugin:@typescript-eslint/recommended", "prettier"],

plugins: ["prettier"],

globals: {

Atomics: "readonly",

SharedArrayBuffer: "readonly",

},

parserOptions: {

ecmaVersion: 2018,

sourceType: "module",

},

rules: {

"prettier/prettier": "error",

"class-methods-use-this": "off",

"no-param-reassign": "off",

camelcase: "off",

"no-unused-vars": ["off"],

"@typescript-eslint/no-unused-vars": "off",

"@typescript-eslint/no-explicit-any": "off",

"@typescript-eslint/ban-types": "off",

"@typescript-eslint/no-var-requires": "off",

"@typescript-eslint/ban-ts-comment": "off",

},

};

## .eslintignore

build/\*

# DNS

## DNS Record Type

* A (Address) Record: Maps a domain name to an IPv4 address.
* AAAA (Quad A) Record: Maps a domain name to an IPv6 address.
* CNAME (Canonical Name) Record: Alias of one name to another. The DNS lookup will continue by retrying the lookup with the new name.
* MX (Mail Exchange) Record: Specifies the mail servers responsible for receiving email for the domain.
* TXT (Text) Record: Can hold arbitrary text. Often used for verification purposes, such as SPF (Sender Policy Framework) records.
* SRV (Service) Record: Specifies the location of servers for specific services.
* PTR (Pointer) Record: Used for reverse DNS lookups, mapping an IP address to a domain name.
* NS (Name Server) Record: Specifies the authoritative DNS servers for a domain.
* SOA (Start of Authority) Record: Provides information about the DNS zone, including the primary name server, email of the domain administrator, domain serial number, and timers for refreshing the zone.
* CAA (Certification Authority Authorization) Record: Specifies which certificate authorities are allowed to issue certificates for the domain.
* NAPTR (Name Authority Pointer) Record: Used for URI (Uniform Resource Identifier) resolution and ENUM (E.164 Number Mapping) services.
* SPF (Sender Policy Framework) Record: Used to define which mail servers are permitted to send email on behalf of a domain.
* DNSKEY (DNS Key Record): Part of DNSSEC, it contains a public signing key.
* DS (Delegation Signer) Record: Used in DNSSEC, it points to DNSKEY records in a child zone.
* TLSA (Transport Layer Security Authentication) Record: Used to associate a TLS server certificate or public key with the domain name where the record is found.