# Domain : Web Development(Tech-A-Thon 3.0)

## Problem Statement 1 (Online Compiler)

Create a online compiler in which user should input the code from an code editor of any programming language and the code should be compiled on the server compiler accordingly and return back the compiled output to the client.

### Frontend:

- You need to create a code editor where user can write the code by choosing any programming language (minimum 3 programming language it should support ex. Java, Python, C++, etc.).
- A terminal to view the code output

### Backend:

- Create any programming language compiler (minimum 3 programming language ex. Java, Python, C++, etc.)
- Dockerise compiler server and host on any cloud services (ex. AWS, Heroku, Azure, etc.)

> NOTE: Don't use any third party APIs nor any open source project for compiler.

# About this Repository

We are Semicolonn Stardust; and this is our Solution for Problem Statement 1 *Online Compiler* of the Domain : Web Development.

Our Site is Live here

To view an in-depth about our project here are two videos from our official YouTube Channel.

Presentation Video

Project Demo Video

## Our Tech Stack :

- Node.js
- Express.js
- Handlebars
- HTML
- CSS
- Vanilla JavaScript
- Docker (For Containerization)
- Git (As Version Control System)
- GitHub
- Digital Ocean
- Apache 2
- Nginx(in Docker)
- Namecheap
- Let's Encrypt (For an SSL Certificate)

# For Contributing this Project

- If you wish to contribute to this repository then you can Fork the Repository.
- Then add or change the things you want in the Repo.
- Then send a Pull Request.
- Once we get it we'll review it and Approve or Deny the PR accordingly.
- We may setup a CI/CD Pipeline in future for the project so your PR will get merged instantly if it meets the following conditions :
    - If an Upstream is setup
    - If GitHub says "Able to Merge"

# Basic Practices of Git for running the

# project on localhost

So there are two ways you can proceed and run the project on localhost.

Following is Method 1 which will allow you to setup the project on localhost:5000

First Fork the Repository and then :

```
$ ~ git clone https://github.com/{your-username}/Online-Compiler.git
```

```
$ ~ cd Online-Compiler
```

```
$ ~ git commit -a -m "Inital Commit for Upstream Push"
```

```
$ ~ git push -u origin main
```

In this method you don't have to worry about `git remote add origin`.

Following is Method 2 where you will need to manually add the remote repository and remote upstream repository.

After Forking the Repository :

- Enter your desired directory from a terminal and then :

```
$ ~ git init
```

```
$ ~ git remote add origin https://github.com/{your-username}/Online-Compiler.git
```

```
$ ~ git commit -a -m "Inital Commit for Upstream Push"
```

```
$ ~ git push
```

```
$ ~ git remote add upstream origin https://github.com/semicolon-stardust/Online-Compiler.g
```

```
$ ~ git remote -v
```

The last line with `git remote -v` will help you in understanding that whether the upstream setup was successful or not.

---

# Pulling the Docker Image

- To pull the docker image you can simply :

```
$ ~ docker pull semicolonstardust/floc.c
```

Then to run the Docker Image

```
$ ~ docker run -d -p 80:5000 semicolon/floc.c
```

**OR you can also learn :**

# How to build a Docker Image locally

- Make sure docker is installed in your device.

If you are in a Linux system you can check it by :

```
$ ~ which docker
```

- First you need to build the Docker.

```
$ ~ docker build -t semicolonstardust/floc.c .
```

- Renaming your Local Container(Quality of life Step)

```
$ ~ docker rename {dynamic_old_container_name} semicolon/floc.c
```

- To run the Image that you build.

```
$ ~ docker run -d -p 80:5000 semicolon/floc.c
```

> NOTE : 8080:5000 -> Here 8080 is the forwarded port and 5000 is the port where the app is actually running

---

This Project is fully Dockerized and you can visit our Organization Profile and the Repository on Docker Hub.

# Things git will ignore

```
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
lerna-debug.log*
.pnpm-debug.log*

# Diagnostic reports (https://nodejs.org/api/report.html)
report.[0-9]*.[0-9]*.[0-9]*.[0-9]*.json

# Runtime data
pids
*.pid
*.seed
*.pid.lock

# Directory for instrumented libs generated by jscoverage/JSCover
lib-cov

# Coverage directory used by tools like istanbul
coverage
*.lcov

# nyc test coverage
.nyc_output

# Grunt intermediate storage (https://gruntjs.com/creating-plugins#storing-task-files)
.grunt

# Bower dependency directory (https://bower.io/)
bower_components

# node-waf configuration
.lock-wscript

# Compiled binary addons (https://nodejs.org/api/addons.html)
build/Release

# Dependency directories
node_modules/
jspm_packages/

# Snowpack dependency directory (https://snowpack.dev/)
web_modules/
```

```
# TypeScript cache
*.tsbuildinfo

# Optional npm cache directory
.npm

# Optional eslint cache
.eslintcache

# Optional stylelint cache
.stylelintcache

# Microbundle cache
.rpt2_cache/
.rts2_cache_cjs/
.rts2_cache_es/
.rts2_cache_umd/

# Optional REPL history
.node_repl_history

# Output of 'npm pack'
*.tgz

# Yarn Integrity file
.yarn-integrity

# dotenv environment variable files
.env
.env.development.local
.env.test.local
.env.production.local
.env.local

# parcel-bundler cache (https://parceljs.org/)
.cache
.parcel-cache

# Next.js build output
.next
out

# Nuxt.js build / generate output
.nuxt
dist
```

```
# Gatsby files
.cache/
# Comment in the public line in if your project uses Gatsby and not Next.js
# https://nextjs.org/blog/next-9-1#public-directory-support
# public

# vuepress build output
.vuepress/dist

# vuepress v2.x temp and cache directory
.temp
.cache

# Docusaurus cache and generated files
.docusaurus

# Serverless directories
.serverless/

# FuseBox cache
.fusebox/

# DynamoDB Local files
.dynamodb/

# TernJS port file
.tern-port

# Stores VSCode versions used for testing VSCode extensions
.vscode-test

# yarn v2
.yarn/cache
.yarn/unplugged
.yarn/build-state.yml
.yarn/install-state.gz
.pnp.*


public/ace-editor
public/src-min

.DS_Store
```

# Things docker will ignore

```
**/.classpath
**/.dockerignore
**/.env
**/.git
**/.gitignore
**/.project
**/.settings
**/.toolstarget
**/.vs
**/.vscode
**/*.*proj.user
**/*.dbmdl
**/*.jfm
**/azds.yaml
**/charts
**/docker-compose*
**/Dockerfile*
**/node_modules
**/npm-debug.log
**/obj
**/secrets.dev.yaml
**/values.dev.yaml
README.md
```