eBook Gratuit

APPRENEZ vscode

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#vscode

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: vscode

It is an unofficial and free vscode ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official vscode.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec vscode

Remarques

Cette section fournit une vue d'ensemble de ce qu'est vscode et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans vscode, et établir un lien avec les sujets connexes. La documentation de vscode étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Date de sortie
0.10.1-Extensionbuilders	2015-11-13
0,10,1	2015-11-17
0,10,2	2015-11-24
0.10.3	2015-11-26
0,10,5	2015-12-17
0,10,6	2015-12-19
0,10,7-initiés	2016-01-29
0,10,8	2016-02-05
0,10,8-initiés	2016-02-08
0,10,9	2016-02-17
0,10.10-initiés	2016-02-26
0.10.10	2016-03-11
0,10.11	2016-03-11
0.10.11-initiés	2016-03-11
0.10.12-initiés	2016-03-20
0,10.13-initiés	2016-03-29
0,10.14-initiés	2016-04-04

Version	Date de sortie
0,10.15-initiés	2016-04-11
1.0.0	**2016-04-14**
1.1.0-initié	2016-05-02
1.1.0	2016-05-09
1.1.1	2016-05-16
1.2.0	2016-06-01
1.2.1	2016-06-14
1.3.0	2016-07-07
1.3.1	2016-07-12
1.4.0	2016-08-03
traduction / 20160817.01	2016-08-17
traduction / 20160826.01	2016-08-26
traduction / 20160902.01	2016-09-02
1.5.0	2016-09-08
1.5.1	2016-09-08
1.5.2	2016-09-14
1.6.0	2016-10-10
1.6.1	2016-10-13
traduction / 20161014.01	2016-10-14
traduction / 20161028.01	2016-10-28
1.7.0	2016-11-01
1.7.1	2016-11-03
traduction / 20161111.01	2016-11-12
traduction / 20161118.01	2016-11-19
1.7.2	2016-11-22

Version	Date de sortie
traduction / 20161125.01	2016-11-26
traduction / 20161209.01	2016-12-09
1.8.0	2016-12-14
1.8.1	2016-12-20
traduction / 20170123.01	2017-01-23
traduction / 20172701.01	2017-01-27
1.9.0	2017-02-02
traduction / 20170127.01	2017-02-03
traduction / 20170203.01	2017-02-03
1.9.1	2017-02-09
traduction / 20170217.01	2017-02-18
traduction / 20170227.01	2017-02-27
1.10.0	2017-03-01
1.10.1	2017-03-02
1.10.2	2017-03-08
traduction / 20170311.01	2017-03-11
traduction / 20170317.01	2017-03-18
traduction / 20170324.01	2017-03-25
traduction / 20170331.01	2017-03-31
1.11.0	2017-04-06
1.11.1	2017-04-06
traduction / 20170407.01	2017-04-07
1.11.2	2017-04-13

Examples

Sous Windows

- Téléchargez le programme d' installation de code Visual Studio pour Windows.
- Une fois téléchargé, exécutez le programme d'installation (VSCodeSetup-version.exe). Cela ne prendra qu'une minute.

Par défaut, le code VS est installé sous C: \ Program Files (x86) \ Microsoft VS Code pour un ordinateur 64 bits.

Remarque: .NET Framework 4.5.2 est requis pour le code VS. Si vous utilisez Windows 7, assurez-vous que .NET Framework 4.5.2 est installé.

Conseil: Le programme d'installation ajoutera éventuellement Visual Studio Code à votre% PATH%, vous pouvez donc taper 'code' à partir de la console. pour ouvrir le code VS sur ce dossier. Vous devrez redémarrer votre console après l'installation pour que la modification de la variable d'environnement% PATH% prenne effet.

Sur mac

- Téléchargez le code Visual Studio pour Mac.
- Double-cliquez sur l'archive téléchargée pour développer le contenu.
- Faites glisser Visual Studio Code.app dans le dossier Applications pour le rendre disponible dans le tableau de bord.
- Ajoutez le code VS à votre Dock en cliquant avec le bouton droit sur l'icône et en choisissant Options, Garder dans le Dock.

Vous pouvez également exécuter le code VS à partir du terminal en tapant «code» après l'avoir ajouté au chemin:

- Lancer le code VS
- Ouvrez la palette de commandes (Ctrl + Shift + P) et tapez «shell command» pour trouver la commande Shell: Installez la commande 'code' dans la commande PATH.

Redémarrez le terminal pour que la nouvelle valeur $ PATH prenne effet. Vous pourrez taper 'code'. dans n'importe quel dossier pour commencer l'édition de fichiers dans ce dossier.

Remarque: Si vous avez toujours l'ancien alias de code dans votre fichier .bash_profile (ou équivalent) d'une version antérieure du code VS, supprimez-le et remplacez-le en exécutant la commande Shell: Installez la commande 'code' dans la commande PATH.

Sur Linux

Distributions basées sur Debian et Ubuntu

La méthode la plus simple pour installer des distributions basées sur Debian / Ubuntu consiste à télécharger et à installer le package .deb (64 bits) via le centre du logiciel graphique s'il est disponible ou via la ligne de commande avec:

```
sudo dpkg -i <file>.deb
sudo apt-get install -f # Install dependencies
```

L'installation du package .deb installe automatiquement le référentiel apt et la clé de signature pour permettre la mise à jour automatique à l'aide du mécanisme système standard. Notez que les fichiers binaires 32 bits et .tar.gz sont également disponibles sur la page de téléchargement.

Le référentiel et la clé peuvent également être installés manuellement avec le script suivant:

```
curl https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > microsoft.gpg
sudo mv microsoft.gpg /etc/apt/trusted.gpg.d/microsoft.gpg
sudo sh -c 'echo "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main" >
/etc/apt/sources.list.d/vscode.list'
```

Ensuite, mettez à jour le cache du package et installez le package en utilisant:

```
sudo apt-get update
sudo apt-get install code # or code-insiders for insiders build
```

Distributions basées sur RHEL, Fedora et CentOS

Nous livrons actuellement le code VS stable 64 bits dans un référentiel yum, le script suivant installera la clé et le référentiel:

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
sudo sh -c 'echo -e "[code]\nname=Visual Studio
Code\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\nenabled=1\ngpgcheck=1\ngpgkey=https://pac
> /etc/yum.repos.d/vscode.repo'
```

Ensuite, mettez à jour le cache du paquet et installez le paquet en utilisant dnf (Fedora 22 et supérieur):

```
dnf check-update
sudo dnf install code
```

Ou sur les anciennes versions utilisant yum:

```
yum check-update
sudo yum install code
```

distributions basées sur openSUSE et SLE

Le référentiel yum ci-dessus fonctionne également pour les systèmes openSUSE et SLE, le script suivant installera la clé et le référentiel:

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
sudo sh -c 'echo -e "[code]\nname=Visual Studio
Code\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\nenabled=1\ntype=rpm-
md\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" >
/etc/zypp/repos.d/vscode.repo'
```

Ensuite, mettez à jour le cache du package et installez le package en utilisant:

```
sudo zypper refresh
sudo zypper install code
```

Paquet AUR pour Arch Linux

Il existe un package AUR (Arch User Repository) géré par la communauté pour VS Code.

Installation manuelle du package .rpm Le package .rpm (64 bits) peut également être téléchargé et installé manuellement, mais la mise à jour automatique ne fonctionnera que si le référentiel ci-dessus est installé. Une fois téléchargé, il peut être installé à l'aide de votre gestionnaire de paquets, par exemple avec dnf:

```
sudo dnf install <file>.rpm
```

Notez que les fichiers binaires 32 bits et .tar.gz sont également disponibles sur la page de téléchargement .

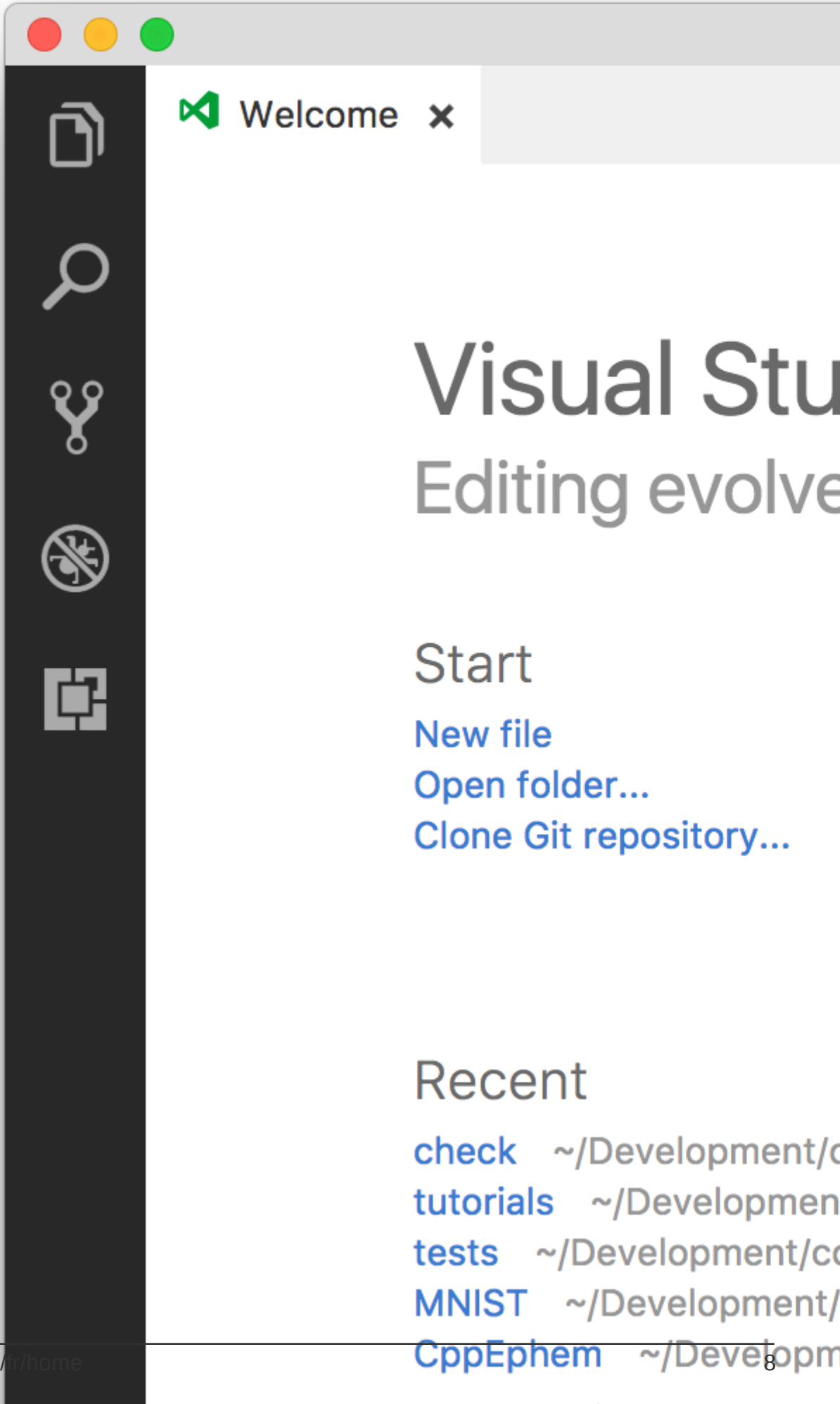
Premiers pas (C ++): HelloWorld.cpp

Le premier programme que l'on écrit généralement dans n'importe quelle langue est le script "hello world". Cet exemple montre comment écrire ce programme et le déboguer à l'aide du code Visual Studio (je vous référerai désormais au code Visual Studio en tant que code VS).

Créer le projet

L'étape 1 consistera à créer un nouveau projet. Cela peut être fait de plusieurs manières. Le premier moyen provient directement de l'interface utilisateur.

1. Programme Open VS Code. Vous serez accueilli avec l'écran d'accueil standard (notez que les images sont prises pendant que vous travaillez sur un Mac, mais qu'elles doivent être similaires à votre installation):


Welcome
Visual Stu
Editing evolve
Start
New file
Open folder...
Clone Git repository...
Recent
check ~/Development/c
tutorials ~/Developmen
tests ~/Development/c
MNIST ~/Development/
CppEphem ~/Developm

. Cela ouvrira une nouvelle fenêtre d'édition où nous pourrons commencer à construire notre script. Allez-y et enregistrez ce fichier (vous pouvez utiliser le menu *Fichier* > *Enregistrer* pour le faire). Pour cet exemple, nous allons appeler le fichier **HelloWorld.cpp** et le placer dans un nouveau répertoire que nous appellerons **VSC_HelloWorld /** .

3. Écrivez le programme. Cela devrait être assez simple, mais n'hésitez pas à copier ce qui suit dans le fichier:

```
#include <iostream>

int main() {
   std::cout << "Hello world!" << std::endl;
   return 0;
}
```

Exécuter le code

Ensuite, nous voulons exécuter le script et vérifier sa sortie. Il y a un certain nombre de façons de le faire. Le plus simple est d'ouvrir un terminal et d'accéder au répertoire que nous avons créé. Vous pouvez maintenant compiler le script et l'exécuter avec gcc en tapant:

```
$ g++ HelloWorld.cpp -o helloworld
$ ./helloworld
Hello World!
```

Oui, le programme a fonctionné! Mais ce n'est pas vraiment ce que nous voulons. Ce serait bien mieux si nous pouvions exécuter le programme depuis VSCode lui-même. Nous avons de la chance si! VSCode dispose d'un terminal intégré auquel on peut accéder via le menu " **Affichage** "> " **Terminal intégré** ". Cela ouvrira un terminal dans la partie inférieure de la fenêtre à partir de laquelle vous pourrez naviguer jusqu'au répertoire **VSC_HelloWorld** et exécuter les commandes ci-dessus.

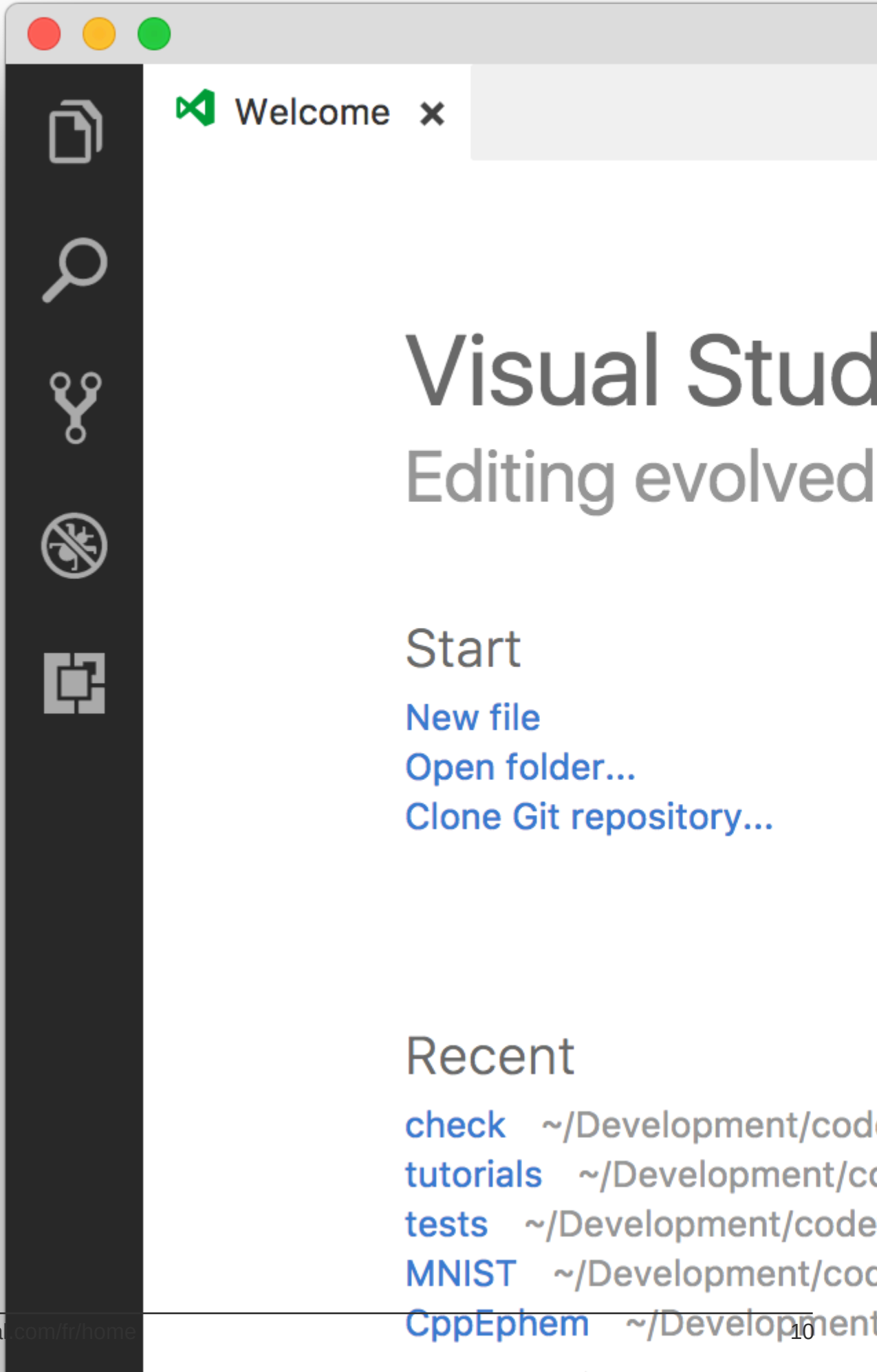
Nous le faisons généralement en exécutant une *tâche d'* exécution. Dans le menu, sélectionnez " **Tâches** "> " **Exécuter la tâche ...** ". Vous remarquerez que vous obtenez une petite fenêtre en haut de la fenêtre avec un message d'erreur (quelque chose comme

Premier programme (C ++): Hello World.cpp

Cet exemple vous présente les fonctionnalités de base de VS Code en démontrant comment écrire un programme "hello world" en C ++. Avant de continuer, assurez-vous que l'extension " **ms-vscode.cpptools** " est installée.

Initialiser le projet

La première étape consiste à créer un nouveau projet. Pour ce faire, chargez le programme VS Code. Vous devriez être accueilli avec l'écran d'accueil typique:


Welcome
Visual Stud
Editing evolved
Start
New file
Open folder...
Clone Git repository...
Recent
check ~/Development/cod
tutorials ~/Development/c
tests ~/Development/code
MNIST ~/Development/co
CppEphem ~/Developmen

" dans l'écran d'accueil. Cela va ouvrir une nouvelle fenêtre de fichier. Allez-y et enregistrez le fichier (" **Fichier** "> " **Enregistrer** ") dans un nouveau répertoire. Vous pouvez nommer le répertoire comme vous le souhaitez, mais cet exemple appelle le répertoire " **VSC_HelloWorld** " et le fichier " **HelloWorld.cpp** ".

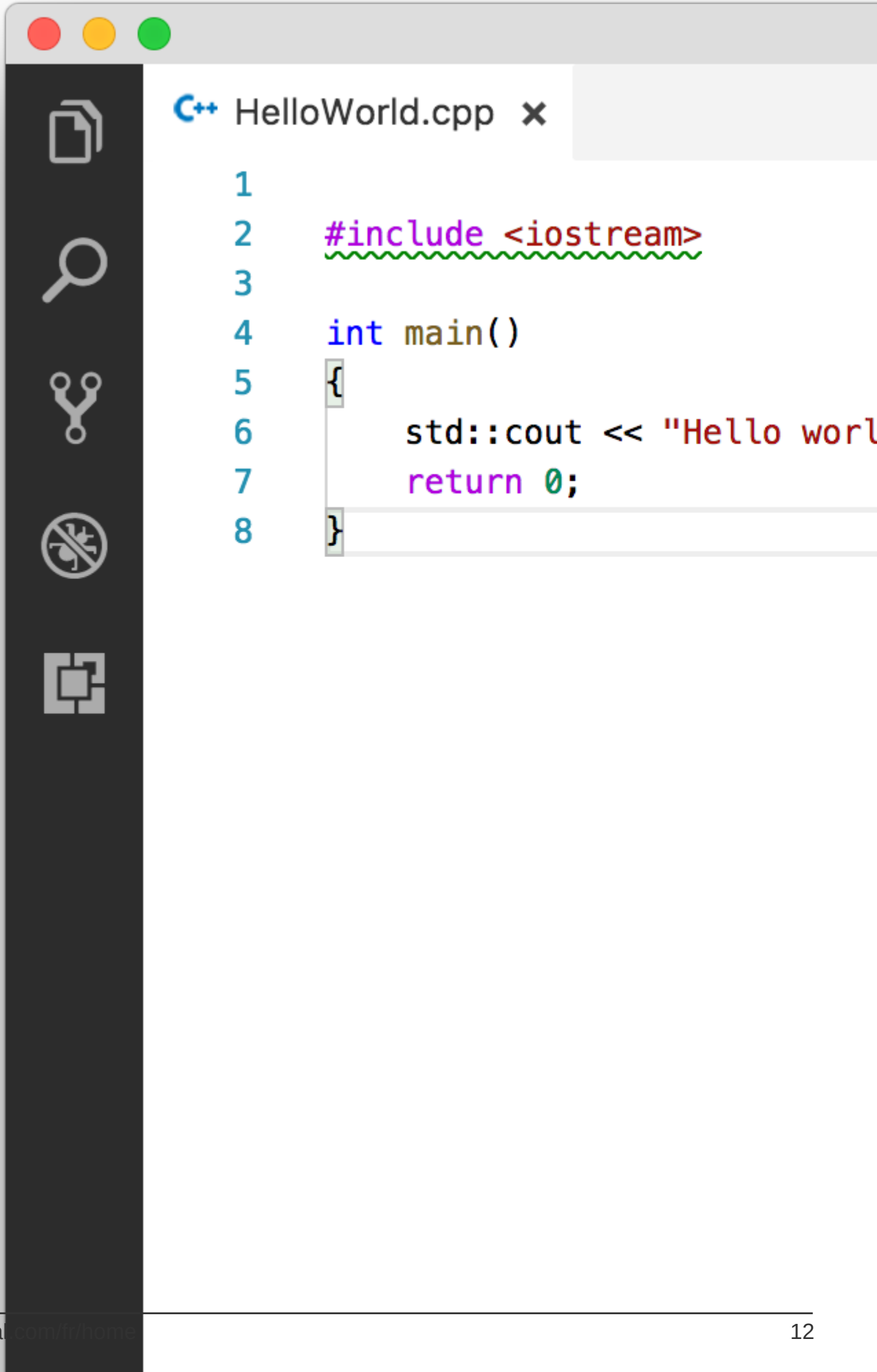
Maintenant, écrivez le programme actuel (n'hésitez pas à copier le texte ci-dessous):

```
#include <iostream>

int main()
{
    // Output the hello world text
    std::cout << "Hello world!" << std::endl;
    return 0;
}
```

Génial! Vous remarquerez également que parce que vous avez installé l'extension " **ms-vscode.cpptools** ", vous avez également une mise en évidence du code. Passons maintenant à l'exécution du code.

Exécuter le script (basique)

Nous pouvons exécuter " **HelloWorld.cpp** " depuis le code VS lui-même. Le moyen le plus simple d’exécuter un tel programme est d’ouvrir le terminal intégré (" **Affichage** "> " **Terminal intégré** "). Cela ouvre une fenêtre de terminal dans la partie inférieure de la vue. De l'intérieur de ce terminal, nous pouvons naviguer dans notre répertoire créé, créer et exécuter le script que nous avons écrit.


HelloWorld.cpp
1
2 #include <iostream>
3
4 int main()
5 {
6 std::cout << "Hello worl
7 return 0;
8 }

Ici, nous avons utilisé les commandes suivantes pour compiler et exécuter le code:

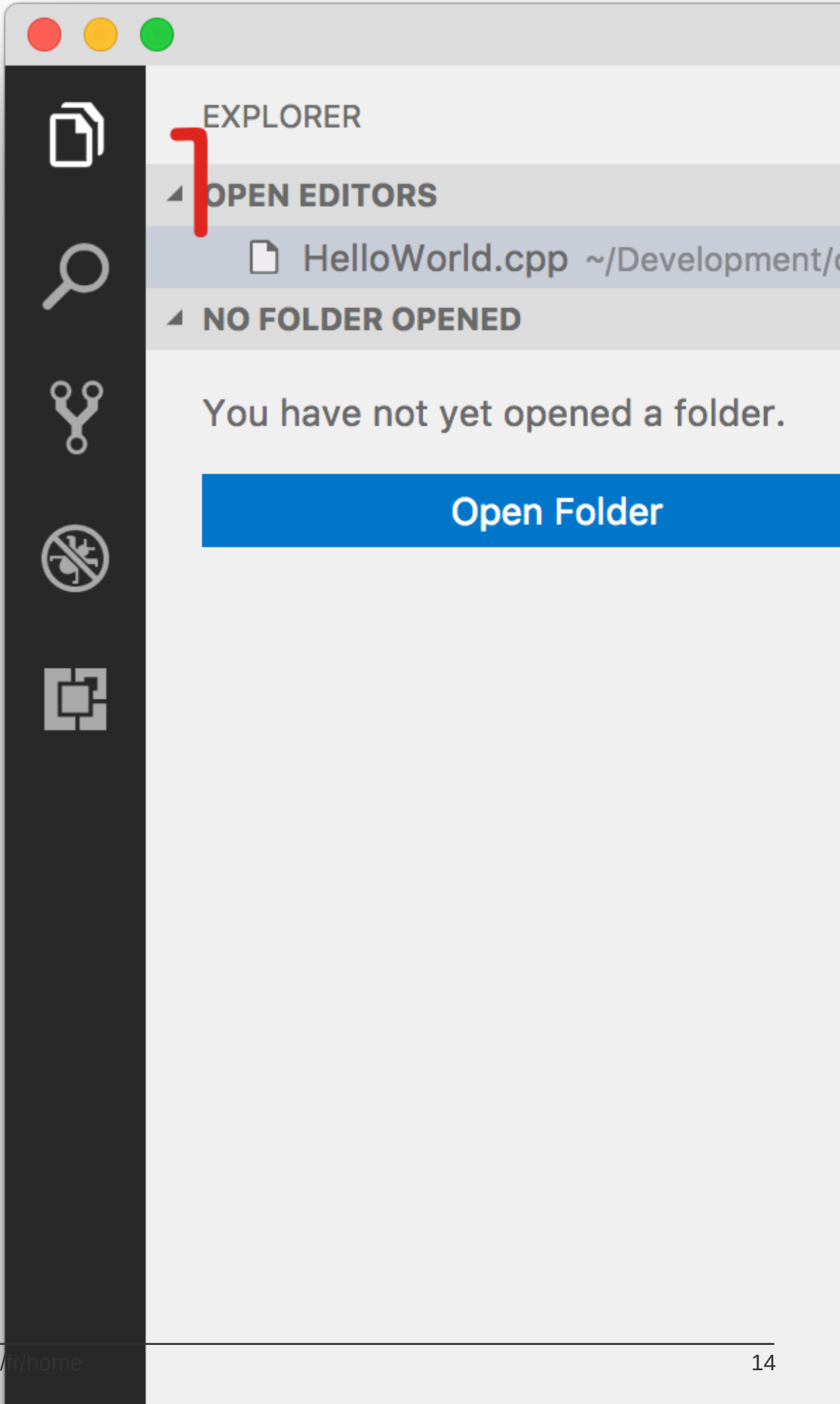
```
$ g++ HelloWorld.cpp -o hellowold
$ ./hellowold
```

Notez que nous obtenons le `Hello World!` attendu `Hello World!` sortie.

Lancer le script (un peu plus avancé)

Génial, mais nous pouvons utiliser le code VS directement pour créer et exécuter le code également. Pour cela, nous devons d'abord transformer le **répertoire** " **VSC_HelloWorld** " en espace de travail. Cela peut être fait par:

1. Ouverture du menu de l' *explorateur* (le plus haut élément du menu vertical à l'extrême gauche)
2. Sélectionnez le bouton *Ouvrir un dossier*
3. Sélectionnez le **répertoire** " **VSC_HelloWorld** " dans **lequel** nous travaillons.


EXPLORER
OPEN EDITORS
HelloWorld.cpp ~/Development/
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder

*Note: Si vous ouvrez un répertoire dans VS Code (en utilisant " **Fichier** "> " **Ouvrir ...** " par exemple), vous serez déjà dans un espace de travail.*

Le menu *Explorer* affiche maintenant le contenu du répertoire.

Ensuite, nous voulons définir les tâches que nous voulons que VS Code exécute. Pour ce faire, sélectionnez " **Tâches** "> " **Configurer la tâche de génération par défaut** ". Dans le menu déroulant, sélectionnez " **Autre** ". Cela ouvre un nouveau fichier appelé " **tasks.json** " qui contient des valeurs par défaut pour une tâche. Nous devons changer ces valeurs. Mettez à jour ce fichier pour contenir les éléments suivants et enregistrez-le:

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "taskName": "build",
            "type": "shell",
            "command": "g++ HelloWorld.cpp -o helloworld"
        },
        {
            "taskName": "run",
            "type": "shell",
            "command": "${workspaceRoot}/helloworld"
        }
    ]
}
```

*Notez que ce qui précède crée également un répertoire **.vscode** caché dans notre répertoire de travail. C'est là que VS Code place les fichiers de configuration, y compris les fichiers de paramètres spécifiques au projet. Vous pouvez en savoir plus sur les tâches ici .*

Dans l'exemple ci-dessus, `${workspaceRoot}` référence au répertoire principal de notre espace de travail, notre **répertoire** " **VSC_HelloWorld** ". Maintenant, pour construire le projet à partir de la méthode, sélectionnez " **Tâches** "> " **Exécuter la tâche de génération ...** " et sélectionnez notre tâche " **build** " créée et " *Continuer sans analyser la sortie de la tâche* " dans les menus déroulants. Ensuite, nous pouvons exécuter l'exécutable en utilisant " **Tasks** "> " **Run Task ...** " et en sélectionnant la tâche " **run** " que nous avons créée. Si vous avez le terminal intégré ouvert, vous remarquerez que "Hello World!" le texte sera imprimé là.

Il est possible que le terminal se ferme avant de pouvoir afficher la sortie. Si cela se produit, vous pouvez insérer une ligne de code comme ceci `int i; std::cin >> i;` juste avant la déclaration de retour à la fin de la fonction `main()` . Vous pouvez ensuite terminer le script en tapant n'importe quel nombre et en appuyant sur <*Entrée*> .

Et c'est tout! Vous pouvez maintenant commencer à écrire et à exécuter vos scripts C ++ à partir du code VS.

Lire Démarrer avec vscode en ligne: https://riptutorial.com/fr/vscode/topic/4489/demarrer-avec-vscode

Chapitre 2: Plusieurs projets mis en place

Remarques

Le projet de tests unitaires mis en place est disponible ici

Examples

Référencement de projets locaux

Les `.sln` et `.proj` .
Au lieu de cela, les **dossiers** sont utilisés dans le code Visual Studio.
Chaque dossier de projet doit avoir un fichier `project.json` distinct.

```
/MyProject.Core
   SourceFile.cs
   project.json

/MyProject.Web
   /Controllers
   /Views
   project.json
```

Pour référencer `MyProject.Core` partir du projet `MyProject.Web` , modifiez le fichier `MyProject.Web\project.json` et ajoutez la dépendance:

```
// MyProject.Web/project.json
{
   "dependencies": {
      "MyProject.Core": {"target": "project"},
     ...
   }
   "buildOptions": {
      "emitEntryPoint": true
   }
}
```

La ligne `"emitEntryPoint": true` indique que `MyProject.Web` est un projet de démarrage pour la solution. Cet indicateur doit être désactivé dans `MyProject.Core` dans son fichier `project.json` :

```
 // MyProject.Core/project.json
 {
   ...
   "buildOptions": {
      "emitEntryPoint": false
   }
}
```

Construire le projet (Mac: ⌘ + `Maj` + `B`, Windows: `Ctrl` + `Maj` + `B)` et chaque projet doit avoir propre `\bin` et `\obj` dossiers avec de nouveaux `.dll` fichiers.

Structure de solution

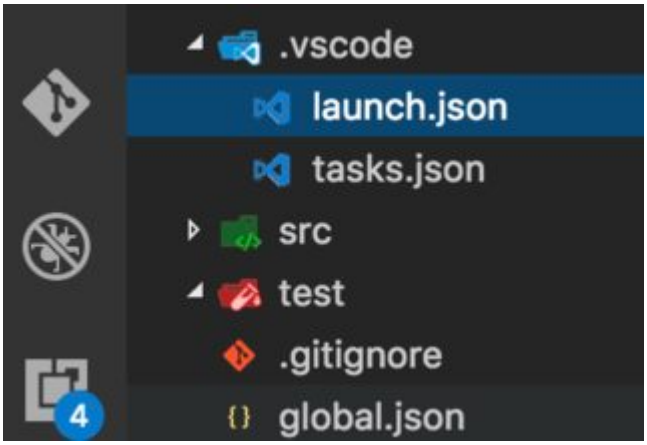
Il est très courant de regrouper des projets, par exemple, placer des projets de test sous le dossier `/test` et les projets sources sous le dossier `/src` . Ajoutez le fichier `global.json` et créez une structure similaire:

```
global.json
/src/
    /MyProject.Core/
       SourceFile.cs
       project.json

    /MyProject.Web/
       /Controllers
       /Views
       project.json

/test/
    /MyProject.Core.UnitTests/
       SourceFileTest.cs
       project.json

    /MyProject.Web.UnitTests/
       /Controllers
       /Views
       project.json
```

Modifiez le fichier `global.json` vide et spécifiez les groupes de projets:

```
{
    "projects":["src", "test"]
}
```

VS Code utilise `tasks.json` pour exécuter des tâches (par exemple, créer une solution) et `launch.json` pour démarrer un projet (par exemple, le débogage). Si vous ne trouvez pas ces fichiers, essayez de démarrer le débogage en appuyant sur `F5` et en ignorant les erreurs, le code VS générera sous le dossier `.vscode` dossier racine avec les fichiers.



Modifiez le fichier `launch.json` et spécifiez le chemin d'accès à votre bibliothèque de démarrage, modifiez `MyProject.Web` avec le nom de votre projet:

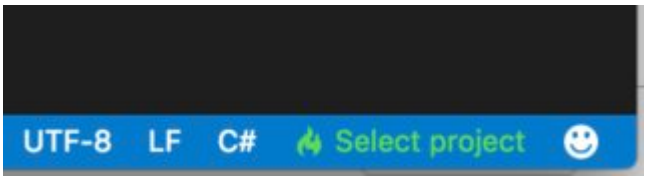
```
{
    "configurations": [
    {
        ...
        "program":
"${workspaceRoot}/src/MyProject.Web/bin/Debug/netcoreapp1.0/MyProject.Web.dll",
        "args": [],
        "cwd": "${workspaceRoot}/src/Washita.Web",
        ...
   }
}
```

Modifiez le fichier `tasks.json` et spécifiez le chemin d'accès à votre bibliothèque de démarrage, modifiez `MyProject.Web` avec le nom de votre projet:

```
{
    "tasks": [
        {
        "taskName": "build",
            "args": [
                "${workspaceRoot}/src/MyProject.Web"
           ],
           "isBuildCommand": true,
           "problemMatcher": "$msCompile"
       }
   ]
}
```

Vous devriez maintenant pouvoir créer et déboguer les fichiers source .NET.

Cependant, Intellisense disparaîtra en raison de la configuration de plusieurs projets. Pour résoudre ce problème, ouvrez un fichier `.cs` et basculez vers le projet approprié (project.json) en `Select project` dans le coin inférieur droit:



Lire Plusieurs projets mis en place en ligne: https://riptutorial.com/fr/vscode/topic/7717/plusieurs-projets-mis-en-place

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec vscode	Community, H. Pauwelyn, Jvinniec, Kronos, RamenChef, Sahan Serasinghe
2	Plusieurs projets mis en place	Artru