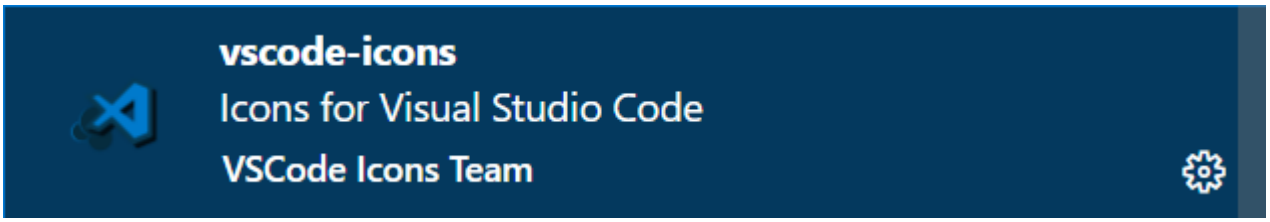
420-KBG : Atelier-1 (Préparé par Nicolas Chourot)

Objectifs

- S'approprier des modules de base de Node.js
- Créer un serveur http embryonnaire
- Utiliser GitHub.com pour conserver une copie de sécurité
- Utiliser https://portal.azure.com pour héberger votre service

---

1) Créez un répertoire *atelier-1* sur un disque physique local à votre PC.
2) Lancez VS Code et installez le greffon suivant :



(Ce greffon permet d'afficher des icones plus compréhensibles)

3) Ouvrez le répertoire *atelier-1* (menu File/Open folder)
4) Créez le fichier *server.js* et entrez le code suivant

```
import http from 'http';
const server = http.createServer((req, res) => {
  console.log(req.url);
});
const PORT = process.env.PORT || 5000;
server.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

5) Ouvrez le panneau du terminal (menu Terminal/New Terminal)
6) Inscrivez la commande suivante : **npm init** et répondez aux questions du formulaire de l'assistant de création du fichier *package.json* :

```
PS D:\CEGEP\Session Automne 2024\KBG\testModule> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (atelier-1) [enter]
version: (1.0.0) [enter]
description: Mon premier server http [enter]
entry point: (server.js) [enter]
test command: [enter]
git repository: [enter]
keywords: [enter]
author: Votre nom [enter]
license: (ISC) [enter]
About to write to D:\CEGEP\Session Automne 2024\KBG\testModule\package.json:

{
  "name": "atelier-1",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "Votre nom",
  "license": "ISC"
}


Is this OK? (yes) [enter]
```

7) Par défaut le type des modules en Node.js est celui du CommonJS. Changez ce type en ajoutant "type":"module" dans le fichier package.json afin de spécifier l'utilisation de modules ES6. Ajoutez aussi la directive qui indique la version de Node.js à utiliser.

```
{
  "name": "atelier-1",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "type": "module",
  "engines": {
    "node": ">20"
  },
  "dependencies": {
    "decode-uri-component": "^0.4.1",
    "filter-obj": "^5.1.0",
    "query-string": "^9.1.0",
    "split-on-first": "^3.0.0"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "Votre nom",
  "license": "ISC"
}
```

Indique que la version du run time de NodeJs devra être supérieur ou égale à 20

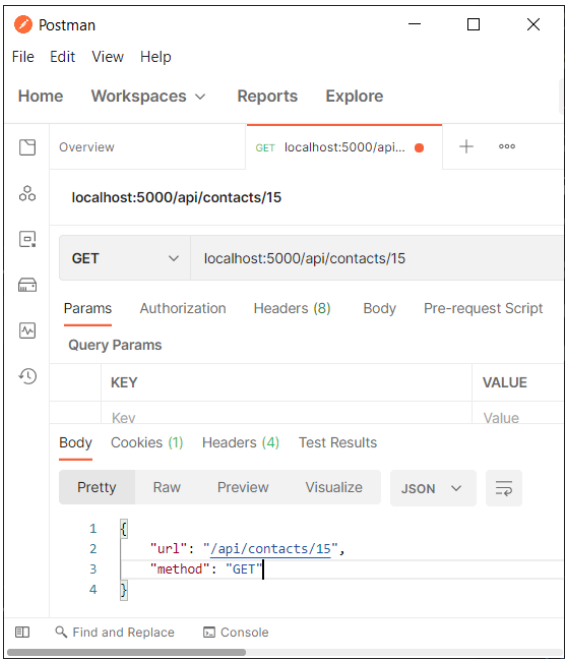
Liste de modules externes nécessaires

8) Ajoutez un fichier de configuration du débogueur : menu Run/Add Configuration… et sélectionnez *Node.js*. Cela ajoutera le fichier *lauch.json* qui servira lors de la prochaine exécution du code se trouvant dans *server.js*
9) Lancez server.js avec le raccourci clavier **F5** et regarder dans la console *DEBUG CONSOLE*. Il devrait être inscrit : *Server running on port 5000*
10) Ouvrez une instance de Google Chrome, et entrez l'url suivant : *localhost:5000/api/contacts/15*
11) Observez l'impact dans *DEBUG CONSOLE* de *VS Code*
12) Arrêtez le server avec le raccourci clavier **SHIFT-F5**
13) Ajoutez le code suivant après l'instruction *console.log(…)* :

```
    let reqInfo = {url:req.url, method:req.method};
    res.writeHead(200, {"Content-Type": "application/json"});
    res.end(JSON.stringify(reqInfo));
```

14) Testez à nouveau (ajoutez si vous voulez le greffon JSON Formatter à votre Google Chrome) https://chrome.google.com/webstore/detail/json-formatter/bcjindcccaagfpapjjmafapmmgkkhgoa
15) Lancez l'application POSTMAN (téléchargez et installez si pas déjà fait)
16) Ajoutez-y une requête GET et réglez l'URL à *localhost:5000/api/contacts/15*

17) Lancez la requête et observez le résultat.



18) Ajoutez le module *query-string* avec la commande de console : ***npm install query-string***
(Notez les changements dans package.json)

19) Ajoutez dans server.js la référence à ce module :

```
import queryString from "query-string";
```

20) Remplacez le corps du callback (req, res) => {…} par le suivant :

```
console.log(req.url);
let reqInfo = { url: req.url, method: req.method, contentType: req.headers['content-type'] };

res.writeHead(200, { "Content-Type": "application/json" });
if (req.method == 'GET') {
    res.end(JSON.stringify(reqInfo));
} else {
    if (req.method == 'POST') {
        let body = [];
        req.on('data', chunk => {
            body += chunk;
        }).on('end', () => {
            try {
                if (req.headers['content-type'] === "application/json")
                    reqInfo.body = JSON.parse(body);
                else
                if (req.headers['content-type'] === "application/x-www-form-urlencoded")
                    reqInfo.body = queryString.parse(body.toString());
                else
                    reqInfo.body = body.toString();
                res.end(JSON.stringify(reqInfo));
            } catch (error) {
                console.log(error);
            }
        });
    }
}
```

Note : Vous pouvez formatter le code en faisant bouton de droite et appeler « Format Document » ou appliquer l'équivalence clavier **SHIFT+ALT+F**

21) Ajoutez une nouvelle requête dans *POSTMAN* :

Méthode : POST,
URL : localhost:5000/api/contacts,
CONTENT-TYPE : application/json
BODY : {"FirstName": "Kyle", "LastName": "Ross", "Email": "Kyle.Ross@clg.qc.ca" }

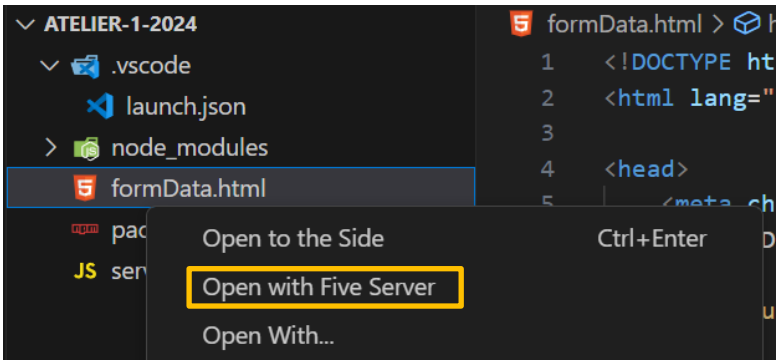
22) Lancez la requête et observez le résultat

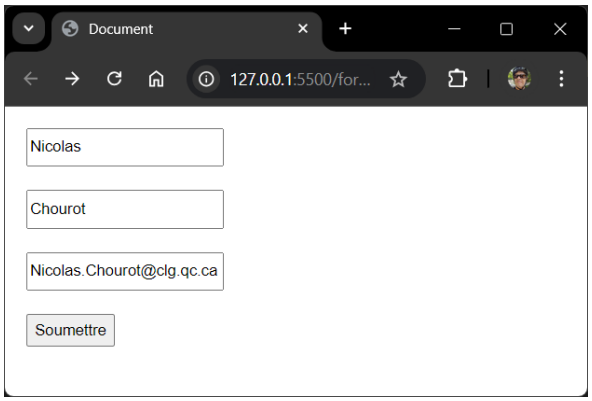
23) Ajoutez un fichier *formData.html* et y collez le contenu suivant :

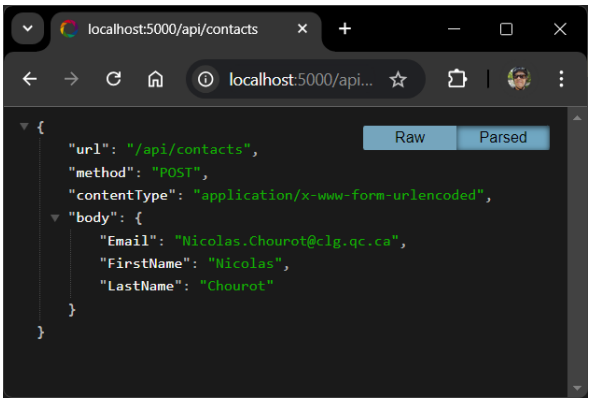
```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        input {
            margin: 10px;
            height: 28px;
        }
    </style>
</head>

<body>
    <form action="http://localhost:5000/api/contacts" method="POST">
        <input type="text" placeholder="Firstname" name="FirstName" required> <br>
        <input type="text" placeholder="Lastname" name="LastName" required><br>
        <input type="email" placeholder="Email" name="Email" required><br>
        <input type="submit" value="Soumettre">
    </form>
</body>

</html>
```

24) Installez les greffons suivants dans VS Code :

25) Lancez la page *formData.html* par l'entremise de ce dernier greffon.

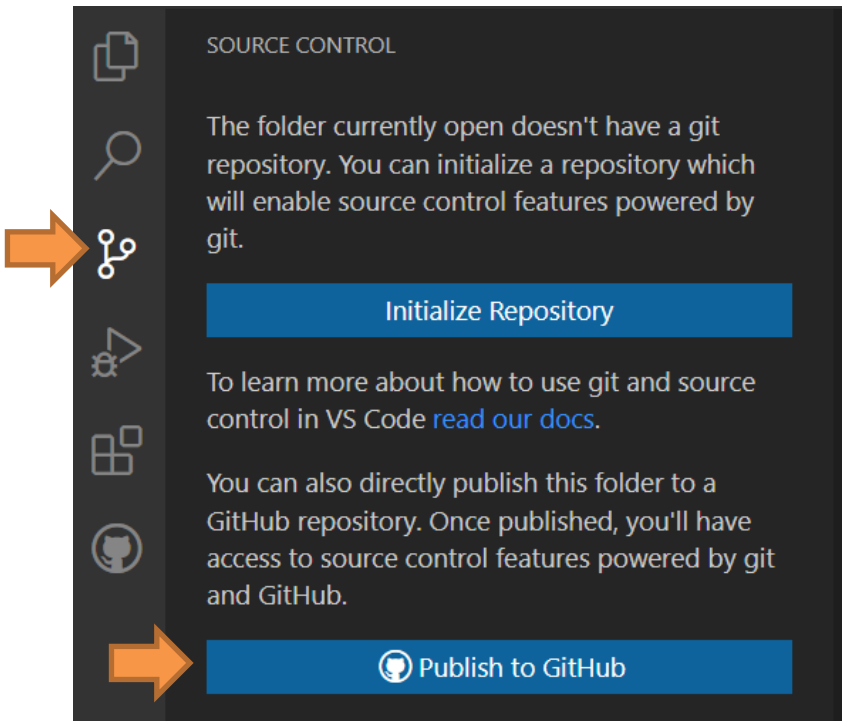




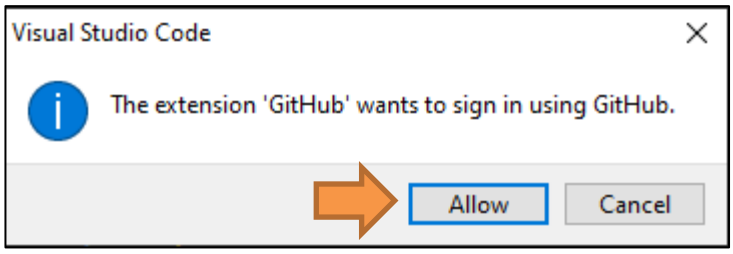


26) Ajoutez dans le projet le fichier .gitignore disponible dans les ressources du cours (Colnet)

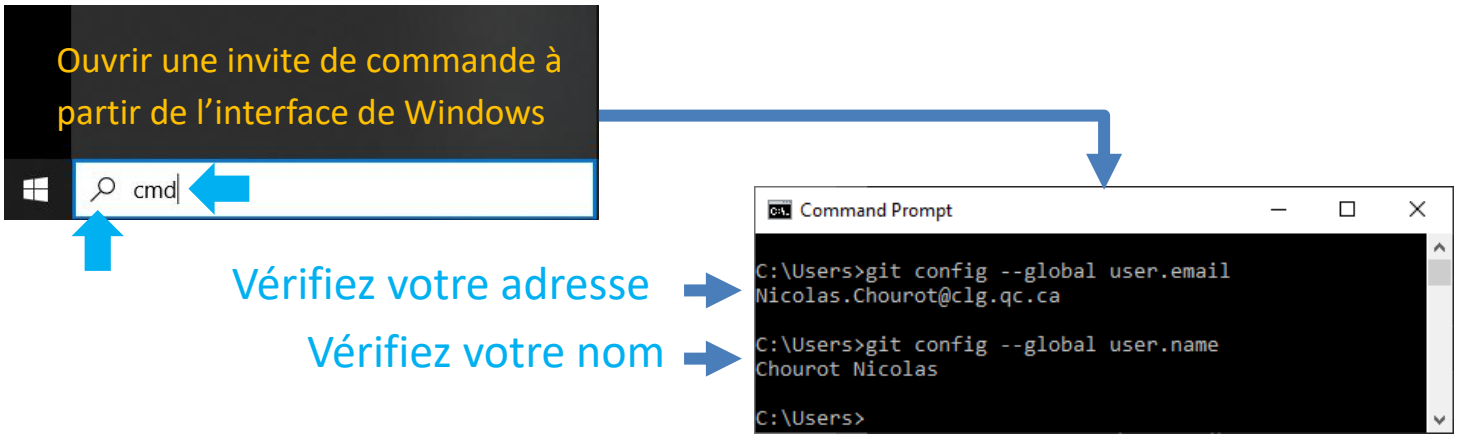
27) Publiez le projet dans GitHub :



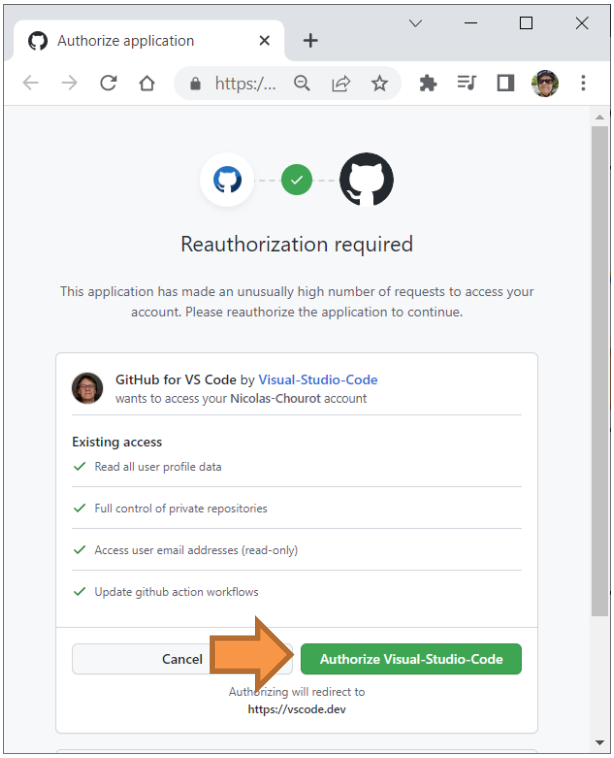
28) Acceptez de se connecter via l'extension GitHub :



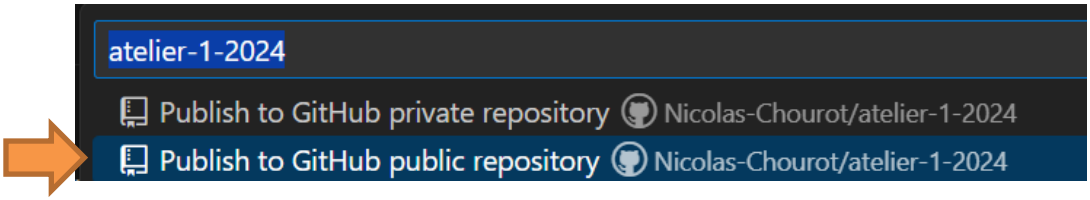
Note : Vérification de configuration d'usager de Git en place sur votre PC avec une invite de commande



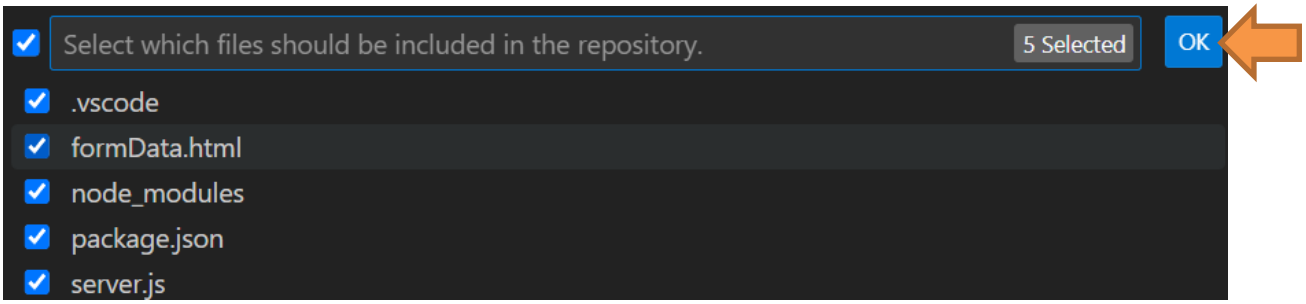
29) Une instance de Google Chrome apparaîtra pour vous demander d'accepter la connexion :



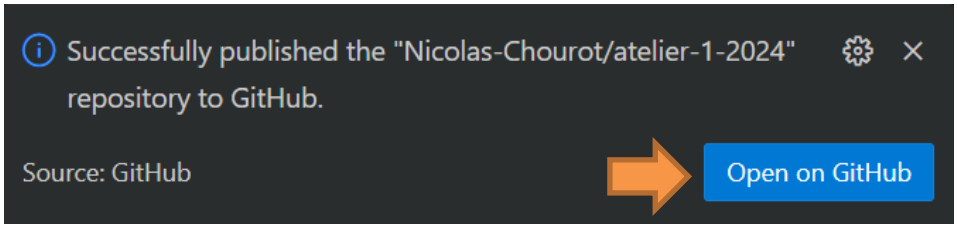
30) Sélectionnez « publier dans un répertoire public » :



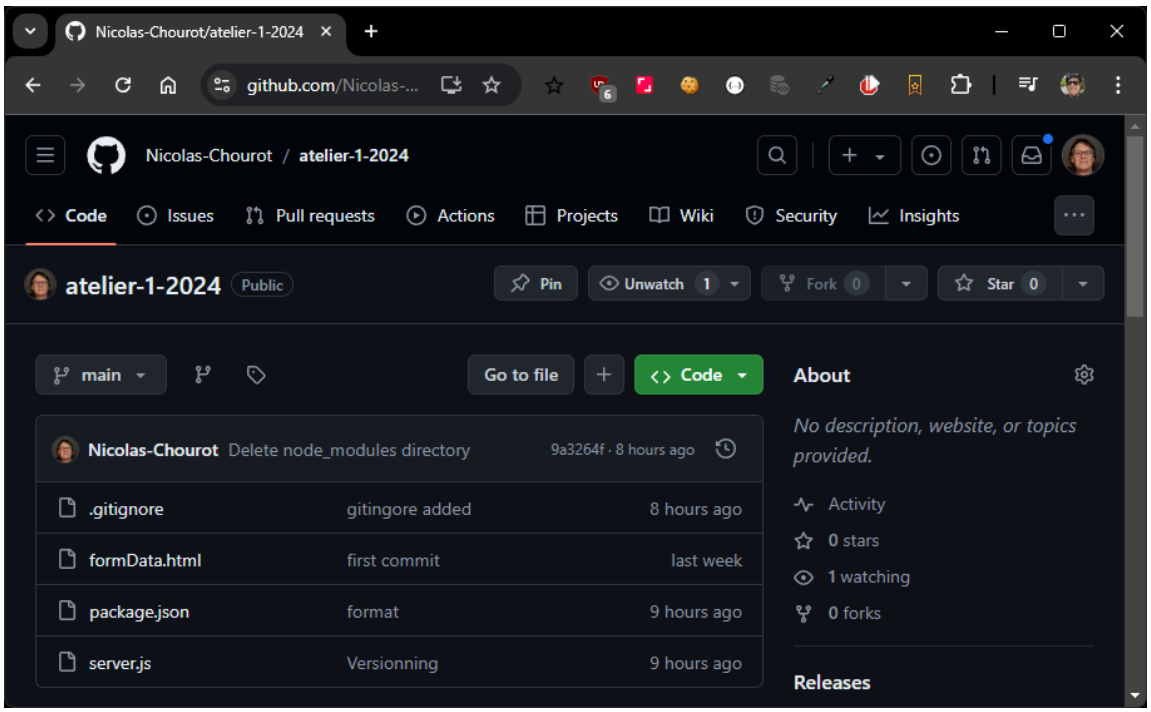
31) Acceptez de publier tous les fichiers et répertoires :



32) Message de confirmation de l'opération, allez dans le site de Github pour contempler votre répertoire :



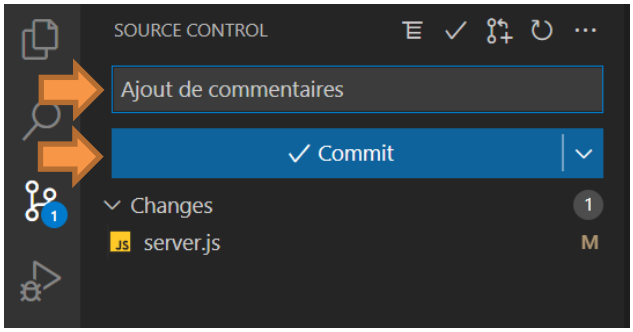
33) Votre répertoire dans GitHub :



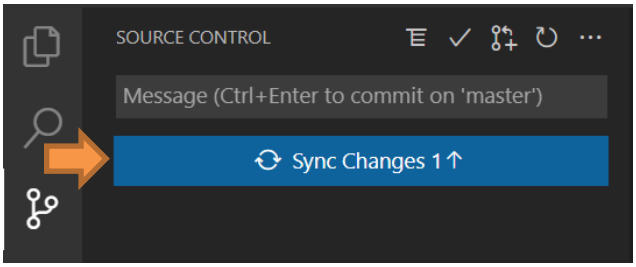
34) Ajouter le commentaire au début du fichier server.js

```
// Mon premier server Http
```

35) Commettre les changements dans Git : Ajouter le commentaire « ajout de commentaire », et commettre :



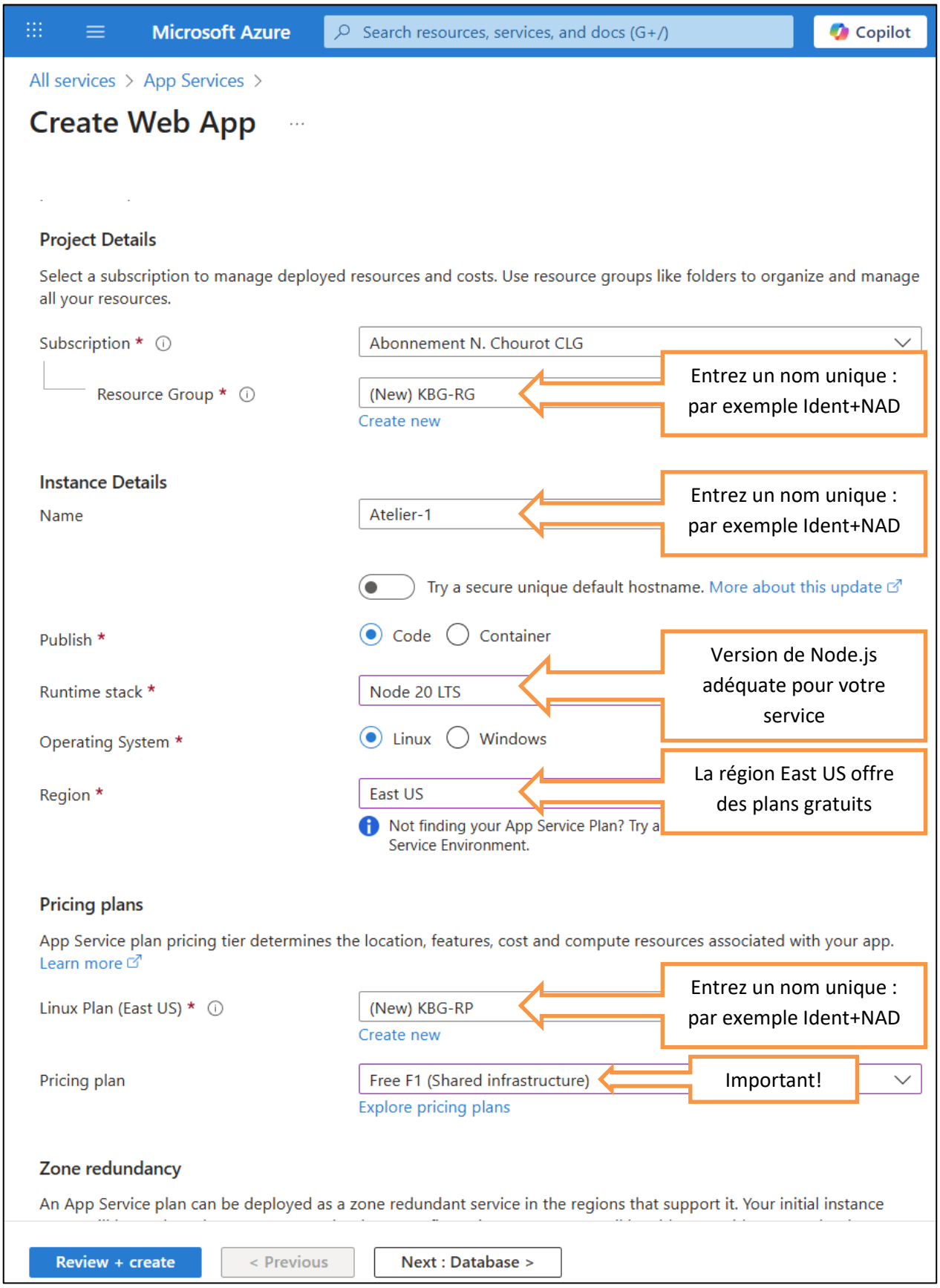
36) Synchroniser les changements dans GitHub :

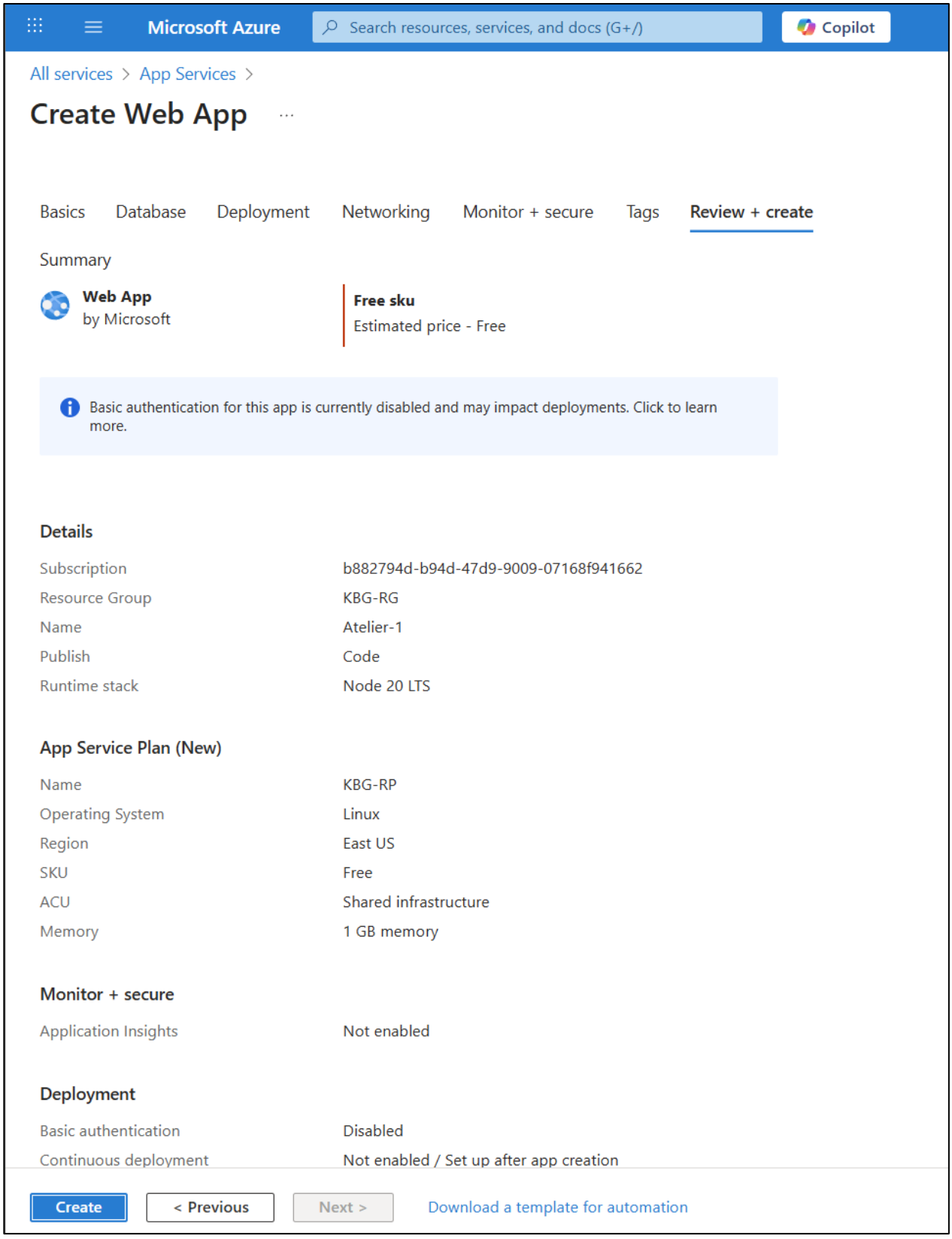


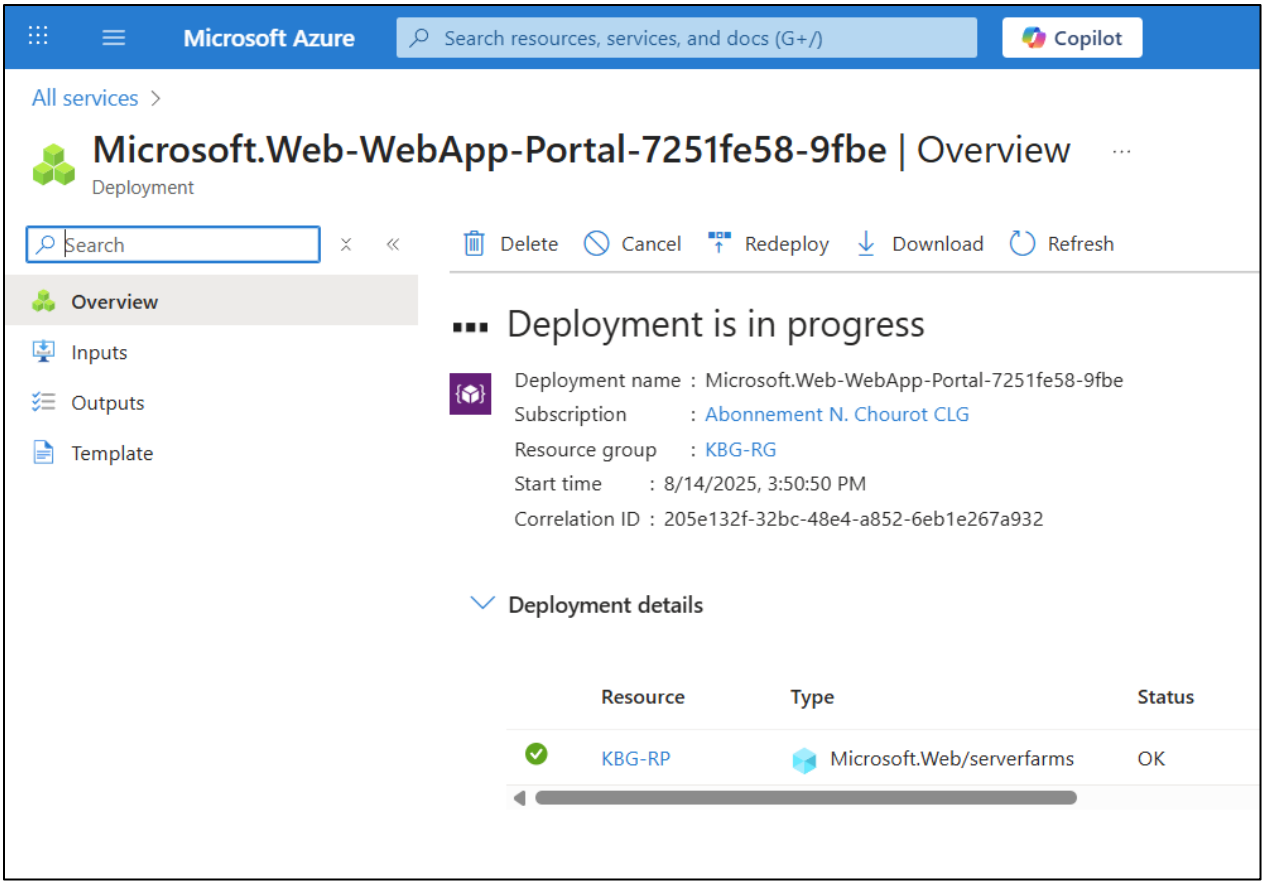
Déploiement sur Azure :

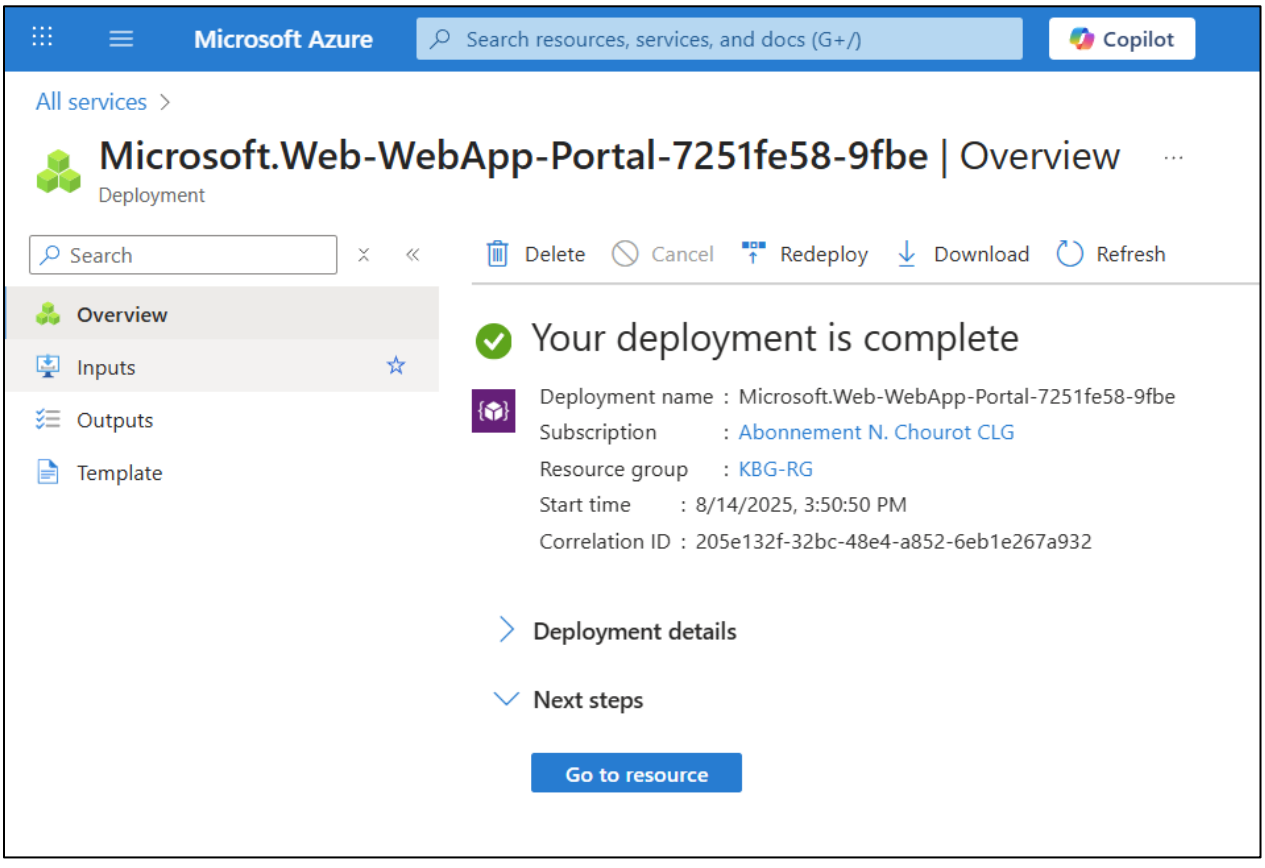
Assurez-vous que vous avez un compte à jour sur Azure. (Voir volet Microsoft dans Colnet)

37) Création d'un Web App dans Azure : (https://portal.azure.com)

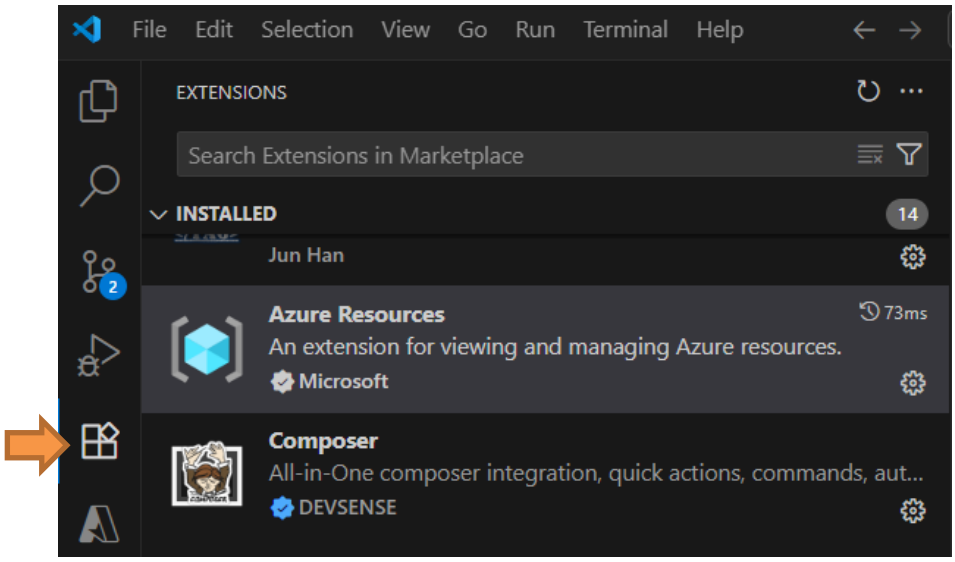



Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
All services > App Services >
Create Web App
Basics
Database
Deployment
Networking
Monitor + secure
Tags
Review + create
Summary
Web App
by Microsoft
Free sku
Estimated price - Free
Basic authentication for this app is currently disabled and may impact deployments. Click to learn more.
Details
Subscription
b882794d-b94d-47d9-9009-07168f941662
Resource Group
KBG-RG
Name
Atelier-1
Publish
Code
Runtime stack
Node 20 LTS
App Service Plan (New)
Name
KBG-RP
Operating System
Linux
Region
East US
SKU
Free
ACU
Shared infrastructure
Memory
1 GB memory
Monitor + secure
Application Insights
Not enabled
Deployment
Basic authentication
Disabled
Continuous deployment
Not enabled / Set up after app creation
Create
< Previous
Next >
Download a template for automation

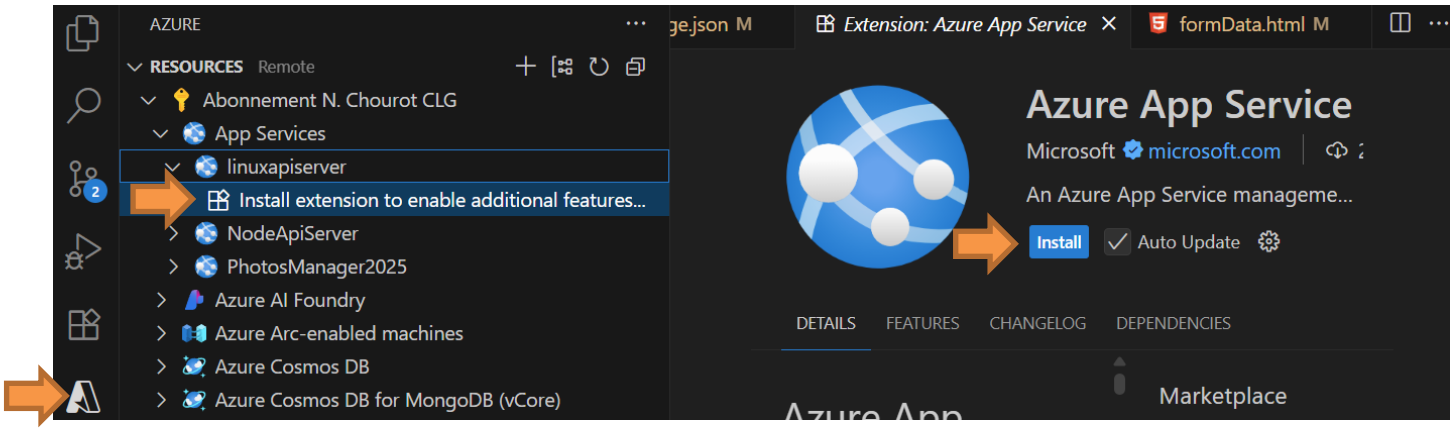

Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
All services
Microsoft.Web-WebApp-Portal-7251fe58-9fbe | Overview
Deployment
Search
Overview
Inputs
Outputs
Template
Delete
Cancel
Redeploy
Download
Refresh
Deployment is in progress
Deployment name : Microsoft.Web-WebApp-Portal-7251fe58-9fbe
Subscription : Abonnement N. Chourot CLG
Resource group : KBG-RG
Start time : 8/14/2025, 3:50:50 PM
Correlation ID : 205e132f-32bc-48e4-a852-6eb1e267a932
Deployment details
Resource
Type
Status
KBG-RP
Microsoft.Web/serverfarms
OK


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
All services
Microsoft.Web-WebApp-Portal-7251fe58-9fbe | Overview
Deployment
Search
Overview
Inputs
Outputs
Template
Delete
Cancel
Redeploy
Download
Refresh
Your deployment is complete
Deployment name : Microsoft.Web-WebApp-Portal-7251fe58-9fbe
Subscription : Abonnement N. Chourot CLG
Resource group : KBG-RG
Start time : 8/14/2025, 3:50:50 PM
Correlation ID : 205e132f-32bc-48e4-a852-6eb1e267a932
Deployment details
Next steps
Go to resource

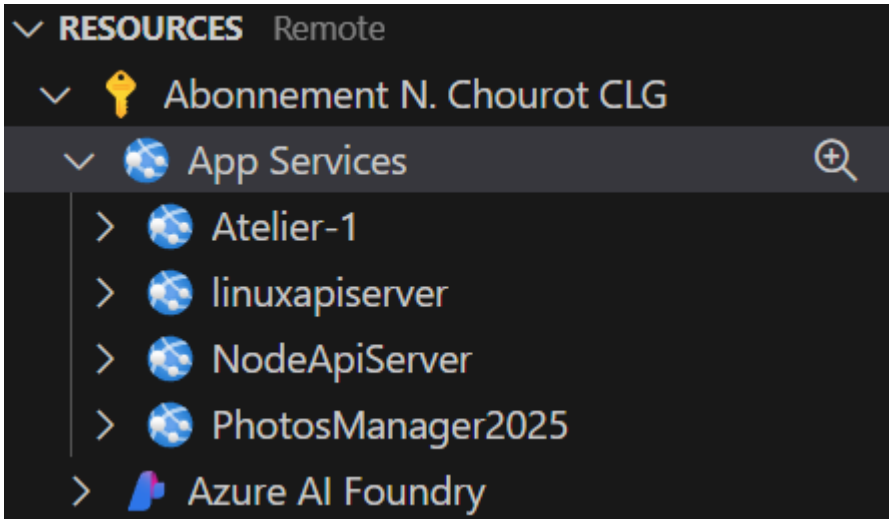
38) Ajout du greffon « Azure Resources » : (il faudra le connecter à votre compte Azure dans le processus)



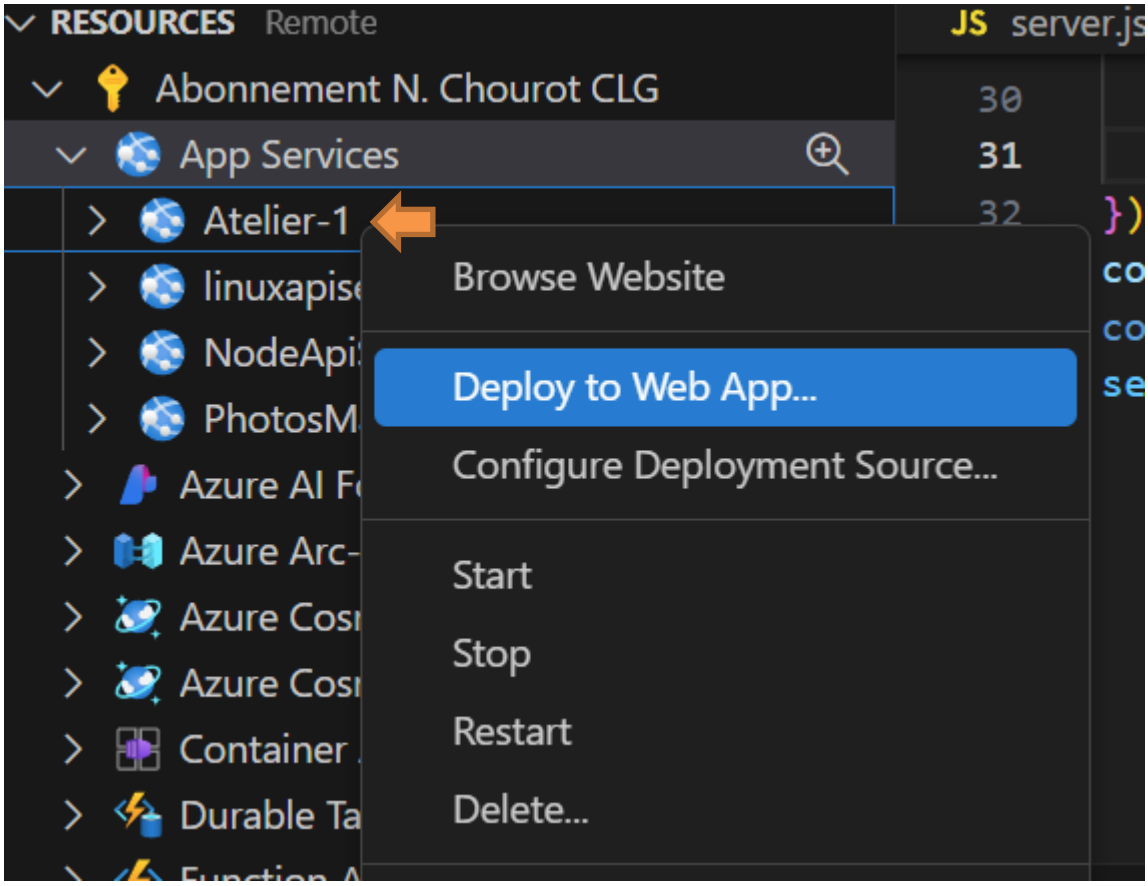
39) Ajout de l'extension Azure App Service :

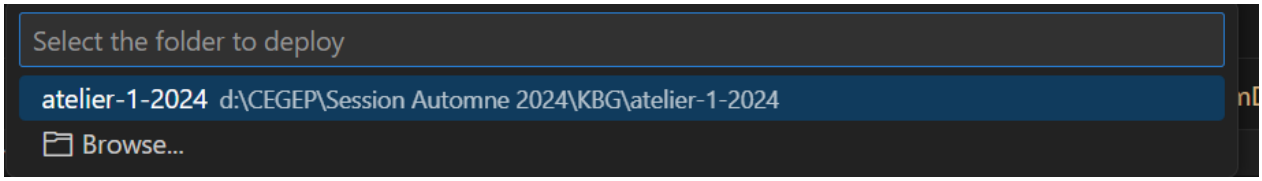


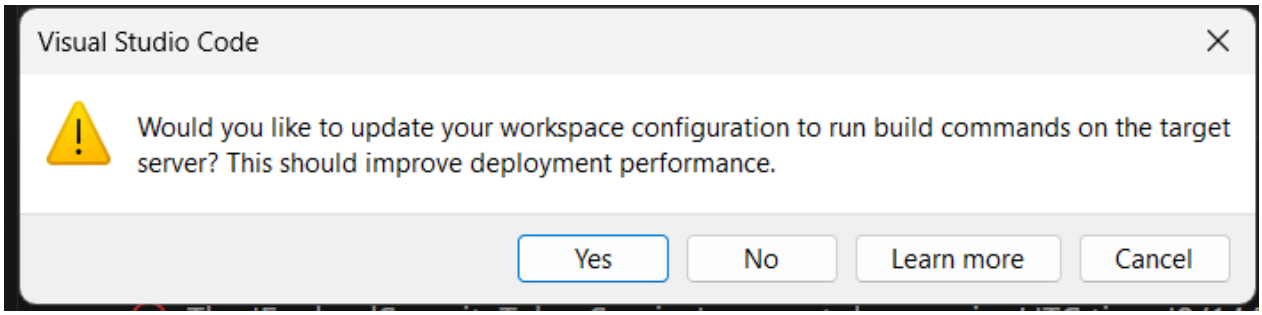
40) Accès au Web App Atelier-1 : (Toutes les entrées déjà en place devront être accessibles)



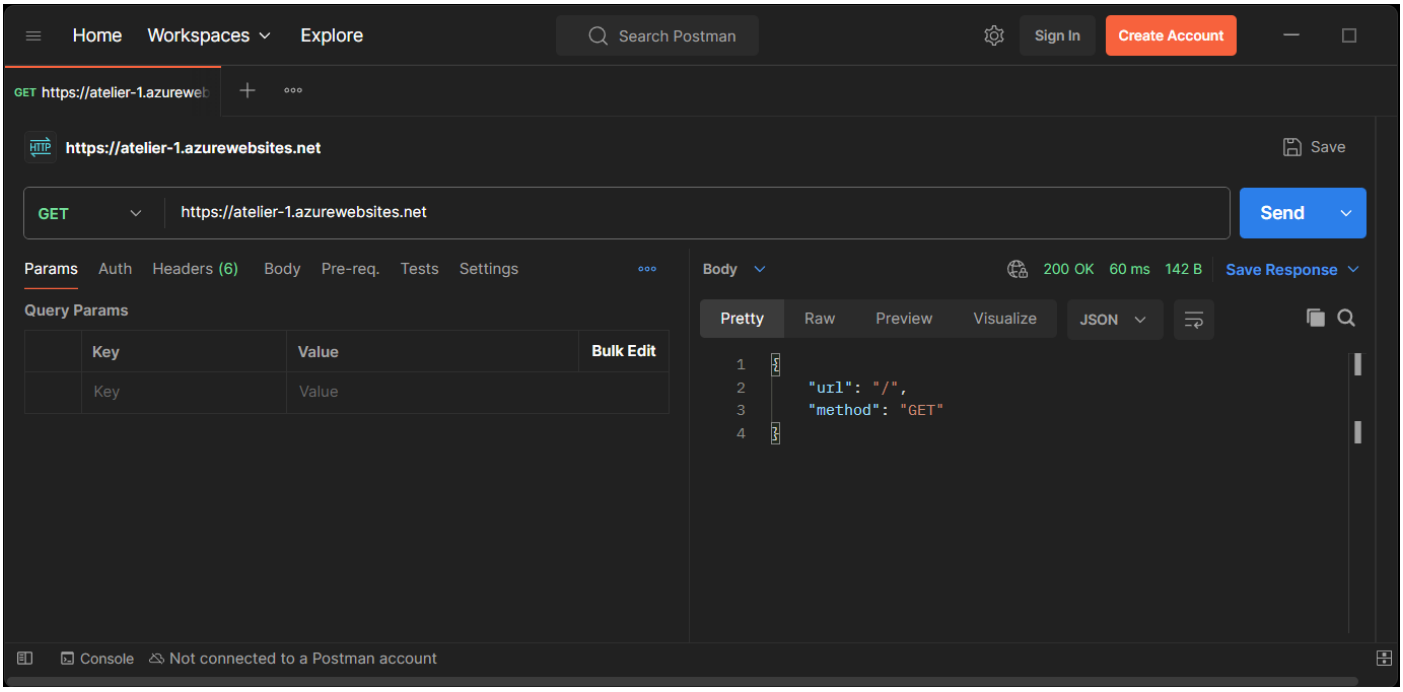
41) Déploiement de votre service : (voir annexe pour l'alternative avec Github)

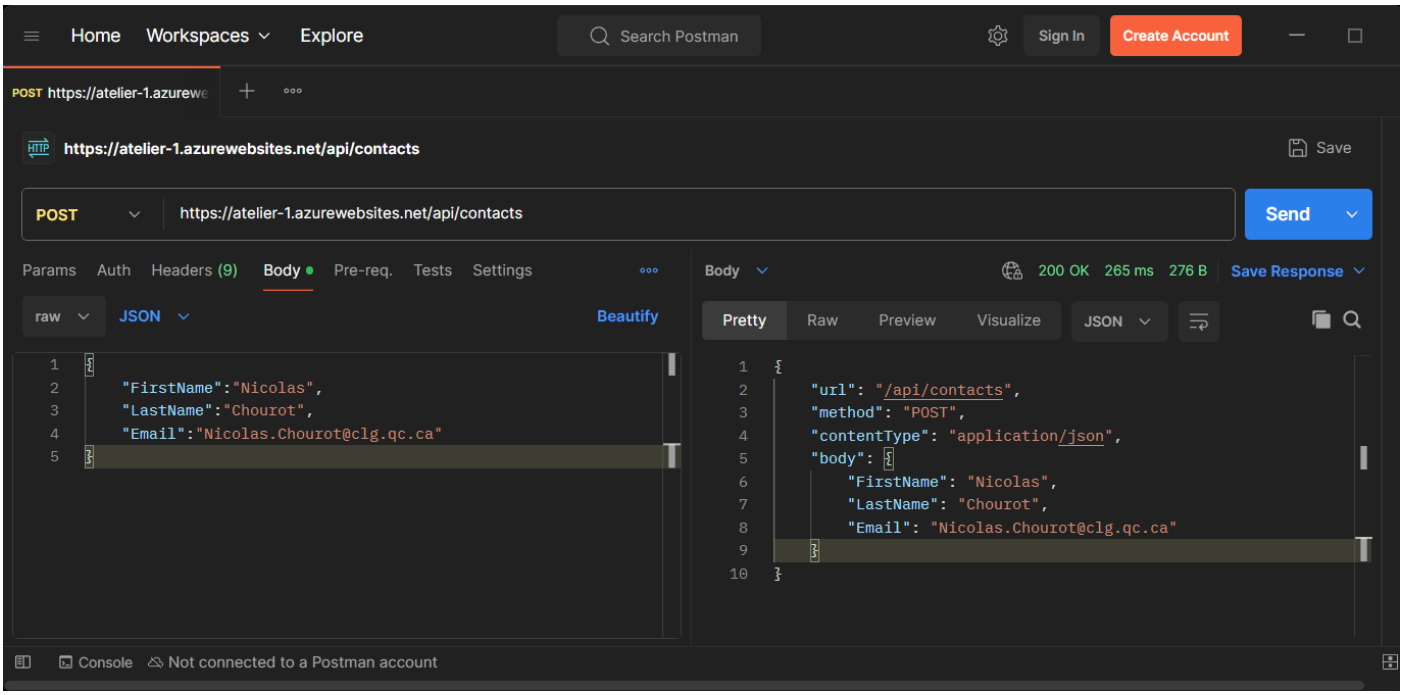


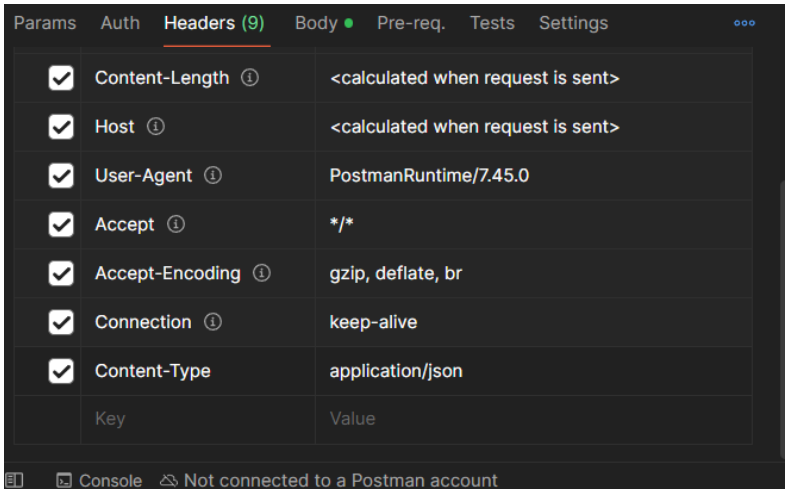




42) Vérifier le bon fonctionnement de votre Azure Web App avec Postman et la version locale de *formdata.html* (utilisez l'url de votre site Glitch plutôt que http://localhost:50000...)



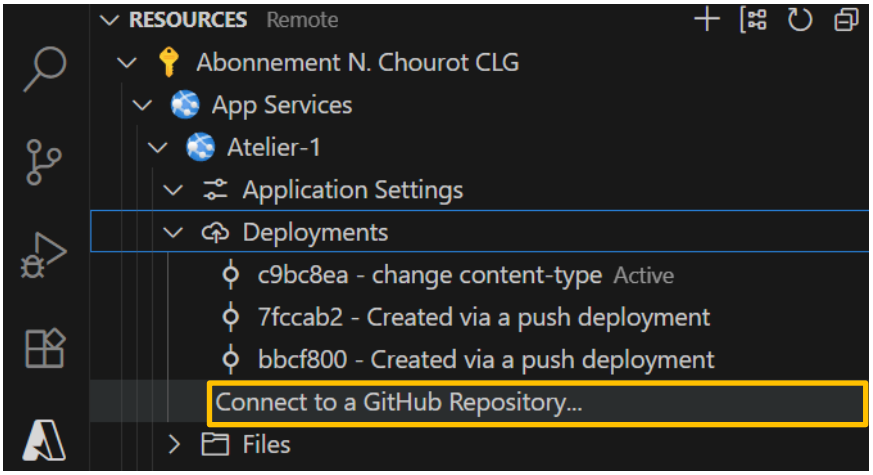


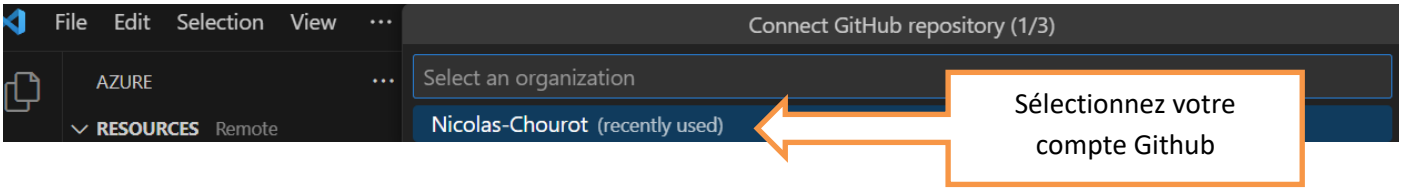


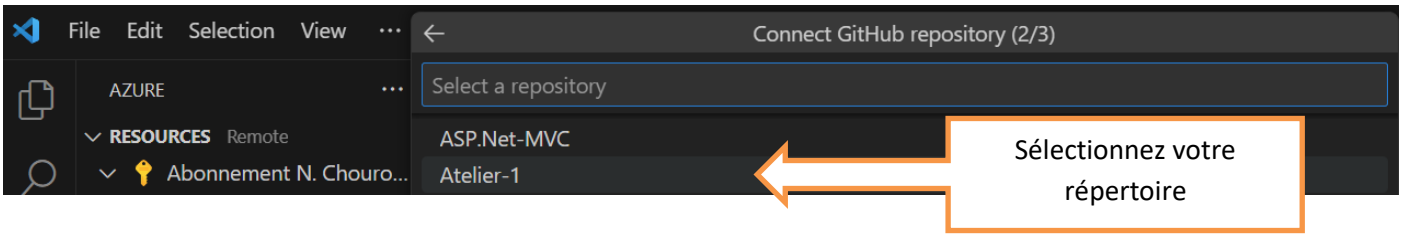
Fin de l’atelier

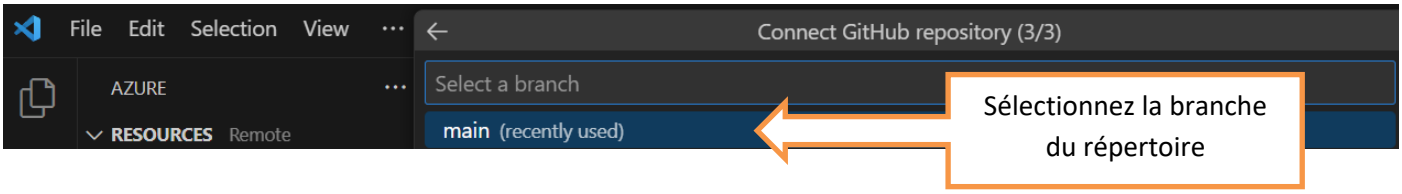
Annexe : Mise en production via Github :

À l'étape 41 de cet atelier choisir :



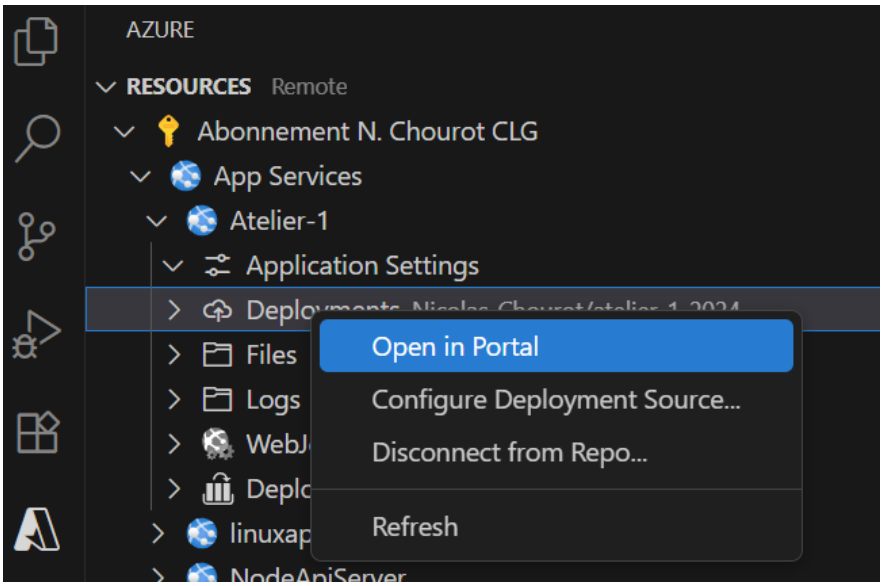


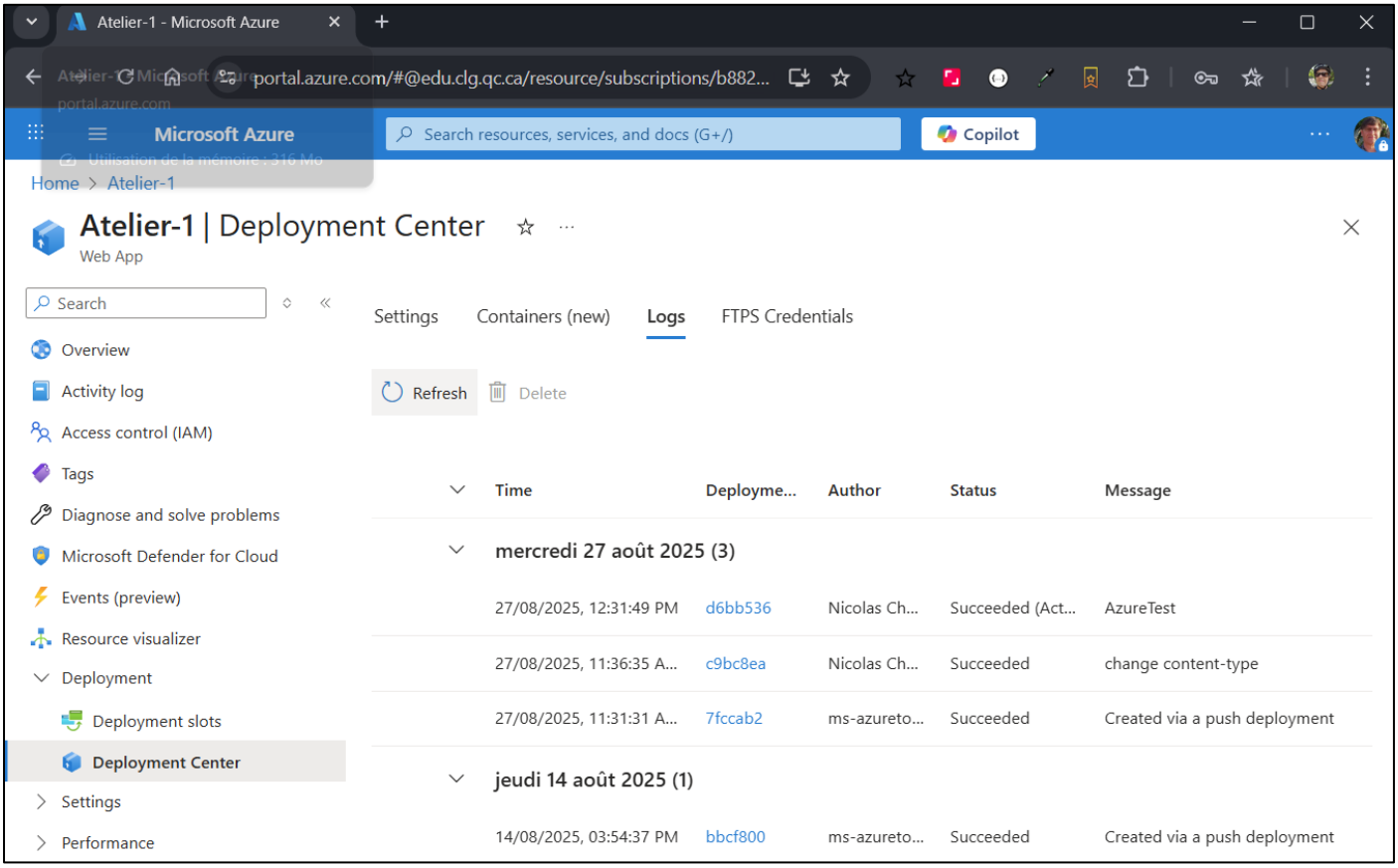




Et voilà! Cependant lors des tests on dirait qu'à chaque push il faut déconnecter et ensuite se reconnecter afin de mettre à jour l'application. À suivre!

Vous pouvez suivre l'historique de déploiement :





Git config :

https://www.google.com/search?q=how+to+set+git+user+in+windows+terminal&rlz=1C1VDKB_frCA1076CA1076&oq=how+to+set+git+user+in+windows+terminal&gs_lcrp=EgZjaHJvbWUyBggAEEUYOdIBCjQ2MTI5ajBqMTWoAgiwAgE&sourceid=chrome&ie=UTF-8

Réglages dans invite de commande :

```
git config --lobal user.name "Your Github account name"

git config --lobal user.email "Your Github account email"

git config --list (to see all the settings)
```