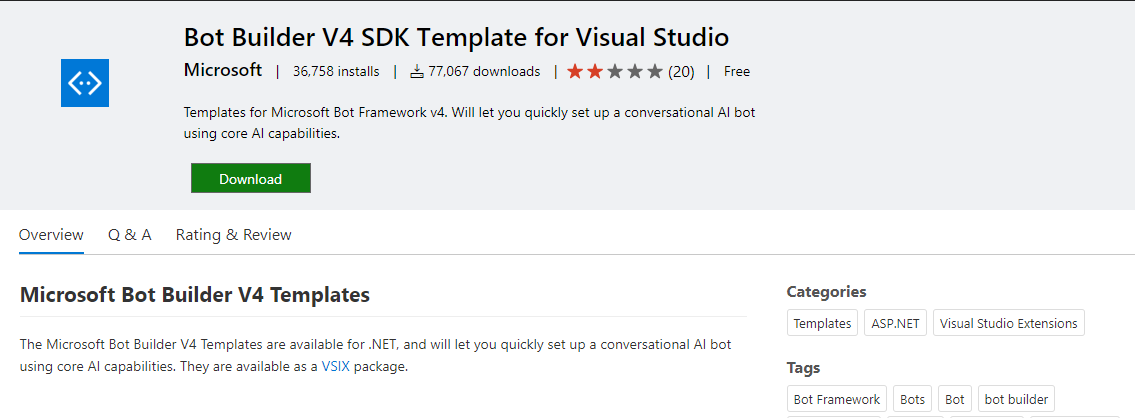
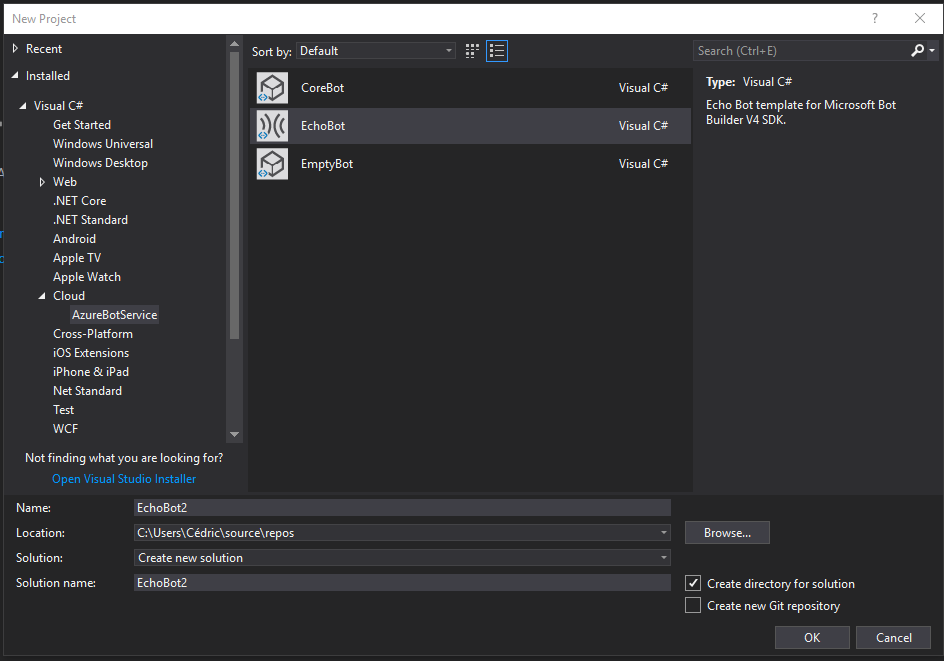
# Créer votre bot avec bot Builder



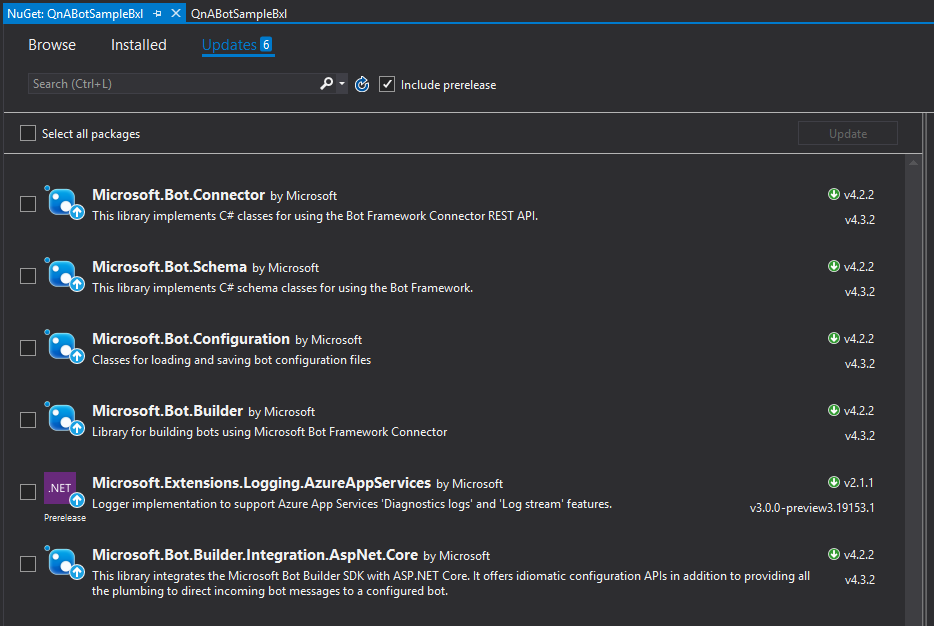
1. Installer Bot Builder V4 SDK Template for Visual Studio

<https://marketplace.visualstudio.com/items?itemName=BotBuilder.botbuilderv4>

1. Créez un nouveau projet de type EchoBot



1. Mettez les package nuget à jour et compilez



1. Exécutez le projet et test du bot.

Maintenant, nous allons tester notre bot pour vérifier s’il fonctionne.

Pour se faire nous allons avoir besoin du bot emulator.

<https://github.com/Microsoft/BotFramework-Emulator/releases/tag/v4.3.3>

1. Nous allons ajouter une base de connaissance.

Pour se faire nous allons créer un service que nous viendrons injecter par injection de dépendance.

* 1. Créez un répertoire Services
  2. Ajouter une interface IQnaMakerServices qui contiendra l’instance du QnAMaker

|  |
| --- |
| public interface IQnaMakerServices  {  QnAMaker QnAMaker { get; }  } |

* 1. Il faut pour ajouter le package nuget du QnAMaker.

|  |
| --- |
| Microsoft.Bot.Builder.AI.QnA |

* 1. Ajoutez une nouvelle class QnaMakerServices qui contiendra la définition du service. Cet object va initialiser le QnAMaker que notre bot utilisera.

|  |
| --- |
| public class QnaMakerServices : IQnaMakerServices  {  private const string Name = "QnABot";  public QnaMakerServices(BotConfiguration config)  {  QnAMaker = Init(config);  }  public QnAMaker QnAMaker { get; }  private QnAMaker Init(BotConfiguration config)  {  var service = config.Services.FirstOrDefault(s => s.Type == ServiceTypes.QnA && s.Name == Name);  var qna = (QnAMakerService)service;  if (qna == null)  {  throw new InvalidOperationException("The QnA service is not configured correctly in your '.bot' file.");  }  if (string.IsNullOrWhiteSpace(qna.KbId))  {  throw new InvalidOperationException("The QnA KnowledgeBaseId ('kbId') is required to run this sample. Please update your '.bot' file.");  }  if (string.IsNullOrWhiteSpace(qna.EndpointKey))  {  throw new InvalidOperationException("The QnA EndpointKey ('endpointKey') is required to run this sample. Please update your '.bot' file.");  }  if (string.IsNullOrWhiteSpace(qna.Hostname))  {  throw new InvalidOperationException("The QnA Host ('hostname') is required to run this sample. Please update your '.bot' file.");  }  var qnaEndpoint = new QnAMakerEndpoint()  {  KnowledgeBaseId = qna.KbId,  EndpointKey = qna.EndpointKey,  Host = qna.Hostname,  };  var qnaMaker = new QnAMaker(qnaEndpoint);  return qnaMaker;  }  } |

* 1. Ajoutez l’instance QnaMakerServices dans le conteur IOC.

Pour cela dans le Startup.cs dans la méthode configureServices ajoutez en singleton le service.

|  |
| --- |
| services.AddSingleton<IQnaMakerServices, QnaMakerServices>(); |

* 1. Votre service est maintenant accessible

Par injection via le constructeur de votre bot, vous allez pouvoir récupérer le qnaMakerServices. Et le conserver comme variable private.

* 1. Maintenant dans la méthode OnTurnAsync, vous allez pouvoir l’utiliser.

Si le type de l’activity est bien message :

|  |
| --- |
| var response = await \_qnaMakerServices.QnAMaker.GetAnswersAsync(turnContext);  if (response != null && response.Length > 0)  {  await turnContext.SendActivityAsync(response[0].Answer);  }  else  {  var msg = @"No QnA Maker answers were found. This example uses a QnA Maker Knowledge Base that focuses on smart light bulbs.  To see QnA Maker in action, ask the bot questions like 'Why won't it turn on?' or 'I need help'.";  await turnContext.SendActivityAsync(msg);  } |

* 1. Ajouter dans le fichier config .bot le point d’entrée du service qna

|  |
| --- |
| {  "type": "qna",  "endpointKey": "cfbbaee1-0973-48cc-b366-a1313758c3db",  "hostname": "https://labbxl.azurewebsites.net/qnamaker",  "id": "117",  "kbId": "cdcd7b37-4d3f-4a10-ba31-2c46ac7e8ea3",  "name": "QnABot",  "subscriptionKey": ""  } |

1. Testez votre bot.