

How to Install the NAFX Strategy Tester

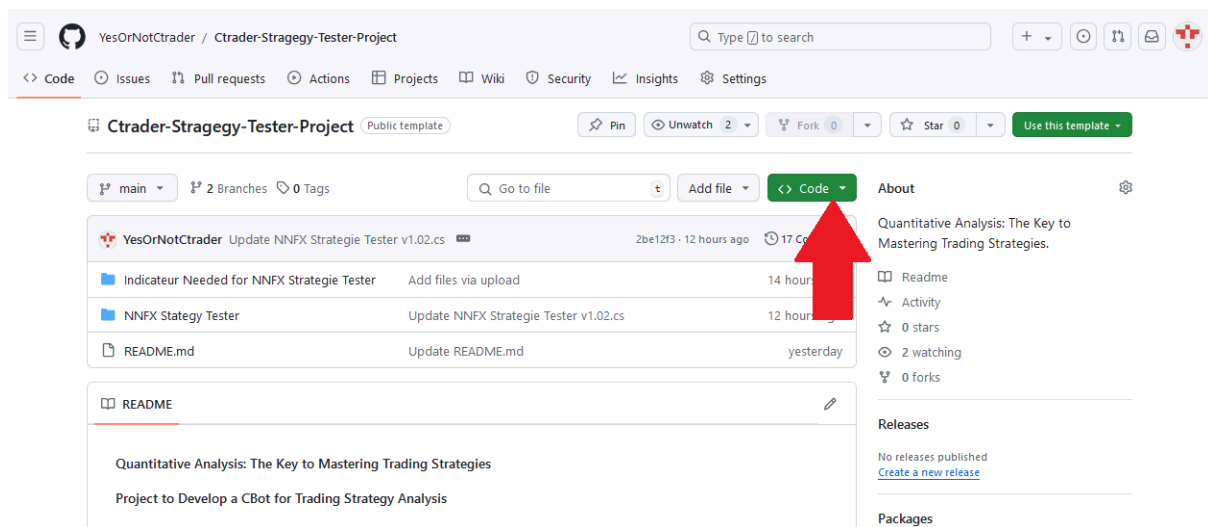
This guide provides two detailed step-by-step methods to install the cBot in the cTrader interface.

II.Download/Install with the .Algo file (Most simple way):

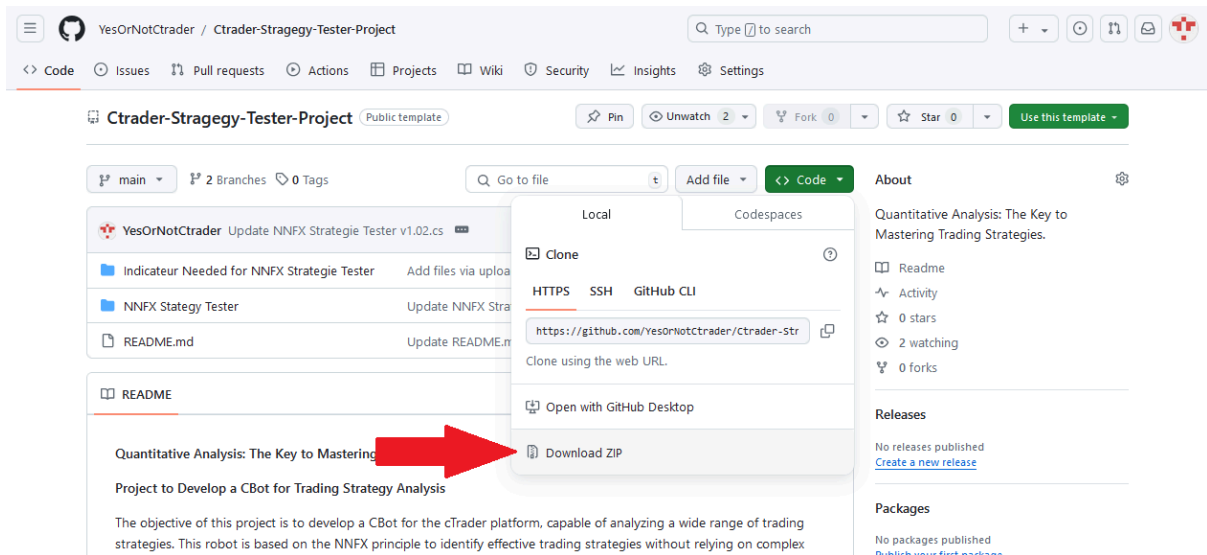
1. Go to the latest available version of the cTrader Strategy Tester.
2. Right-click on the file.
3. Save all sources, indicators, and robots.
4. Go to your download folder.
5. Double-click the downloaded files.
6. Automatic installation by cTrader.

II.Download/Install with the ZIP file:

- 1.
2. Go to the designated page
<https://github.com/YesOrNotCtrader/Ctrader-Stragegy-Tester-Project>
3. Click on the green **Code** button.

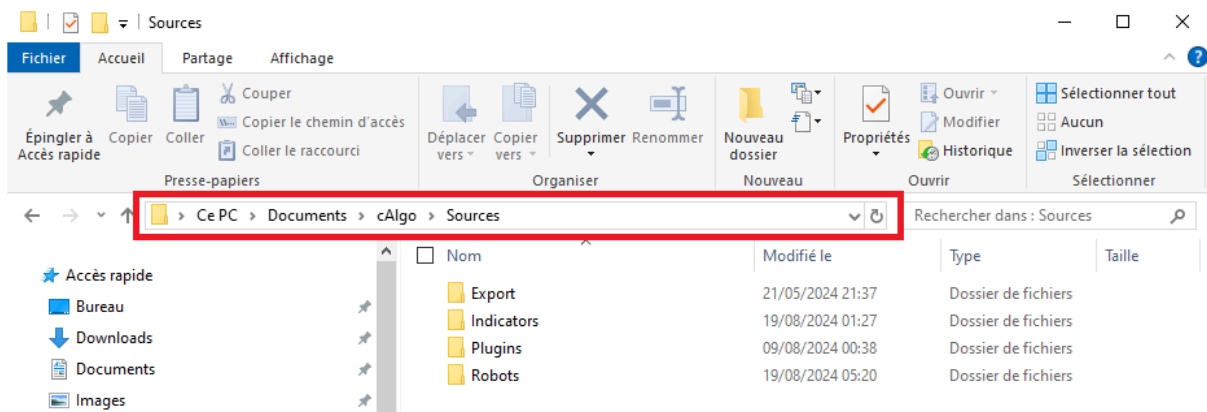


4. Select the **Download ZIP** option to download the file.

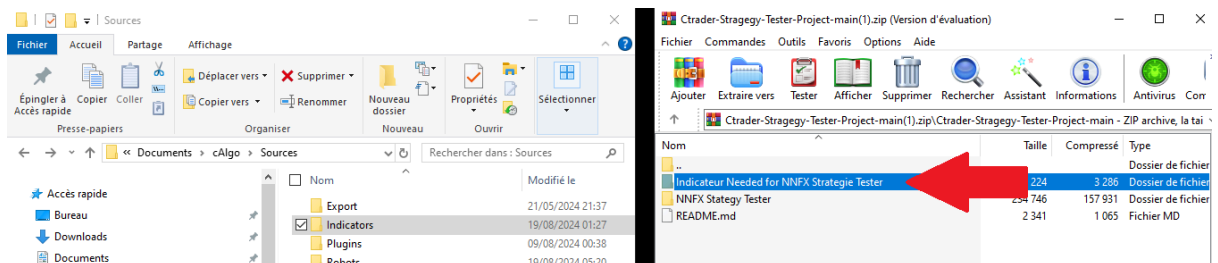


Add the Downloaded Files to the cAlgo Folder:

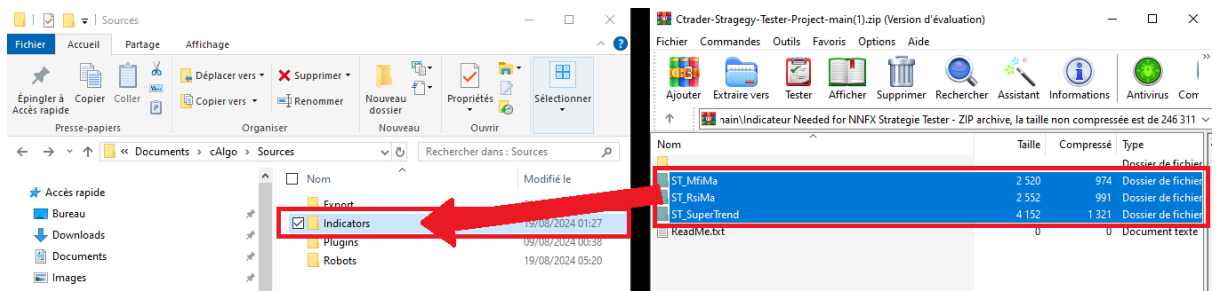
1. Open the **Source** folder for cTrader. This folder is usually located in **Documents** -> **cAlgo** -> **Source**.



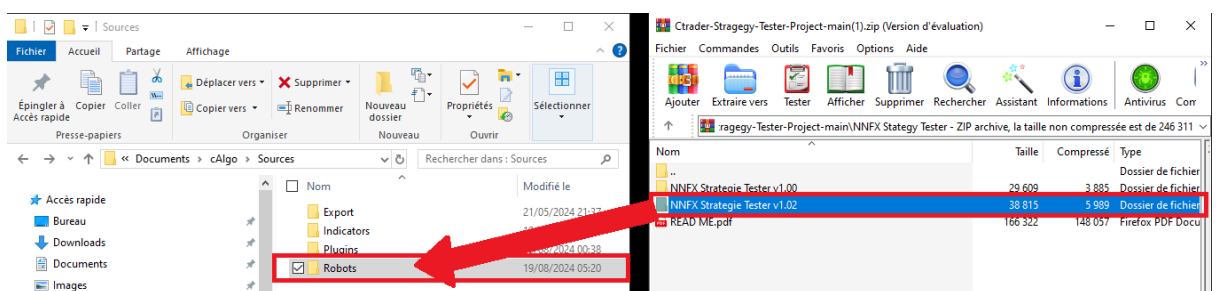
2. Open the downloaded ZIP file and click on "Indicateur Needed For NAFX Strategie Tester".



3. Select all folders named **"ST_..."** (ctrl + click) and Drag and drop these folders into the **Indicators** folder located within the Source folder (from step 1).



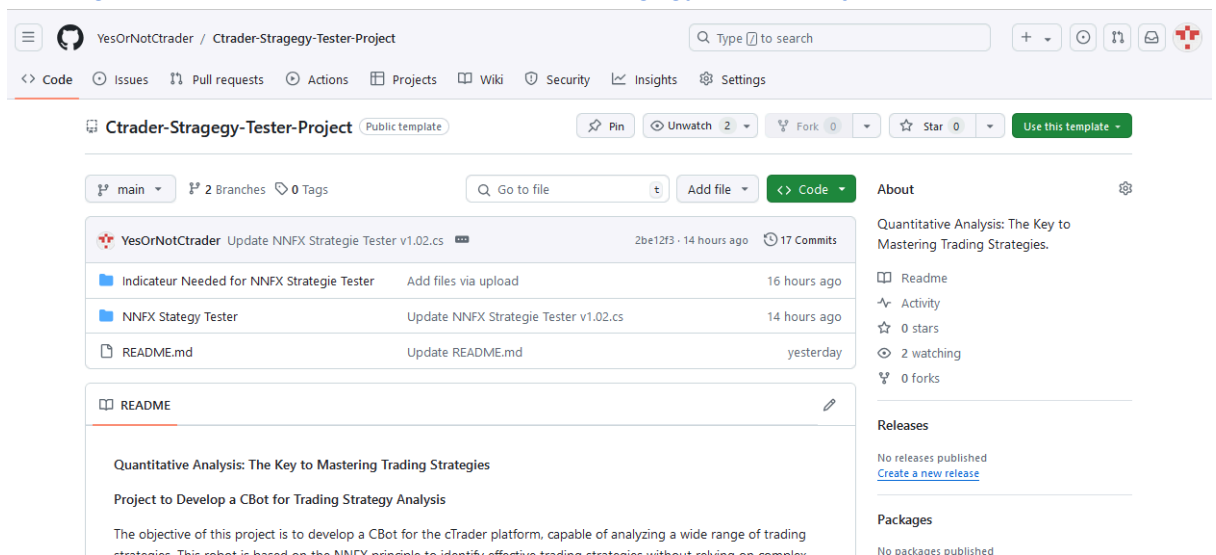
- Return to the **root** of the downloaded folder (click on "..") and open the folder containing the latest version of cbot located in **"NNFX Strategy Tester"** Folder. For this **demonstration, we are using version 1.02** (the higher the number, the newer the version), and drag the downloaded folder into **Robots**.



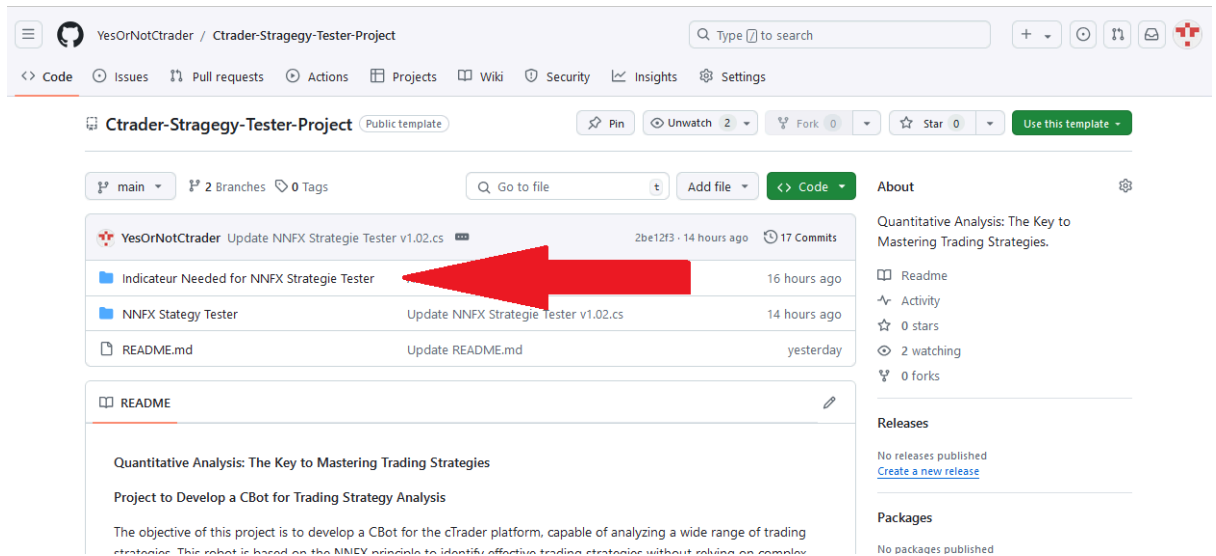
III.Create With “.Cs “ Folder

- Open the GitHub link. :

<https://github.com/YesOrNotCtrader/Ctrader-Strategy-Tester-Project>



2. Click on "Indicators Needed for NNFX Strategy Tester."



The screenshot shows the GitHub repository page for 'CtTrader-Strategy-Tester-Project'. The commit history is displayed, and a red arrow points to the commit 'Indicateur Needed for NNFX Strategie Tester'.

| Commit | Message | Time |
|---|---------------------------------------|--------------|
| 2be12f3 | Update NNFX Strategie Tester v1.02.cs | 14 hours ago |
| 17 | Commits | |
| Indicateur Needed for NNFX Strategie Tester | | 16 hours ago |
| NNFX Strategie Tester | Update NNFX Strategie Tester v1.02.cs | 14 hours ago |
| README.md | Update README.md | yesterday |

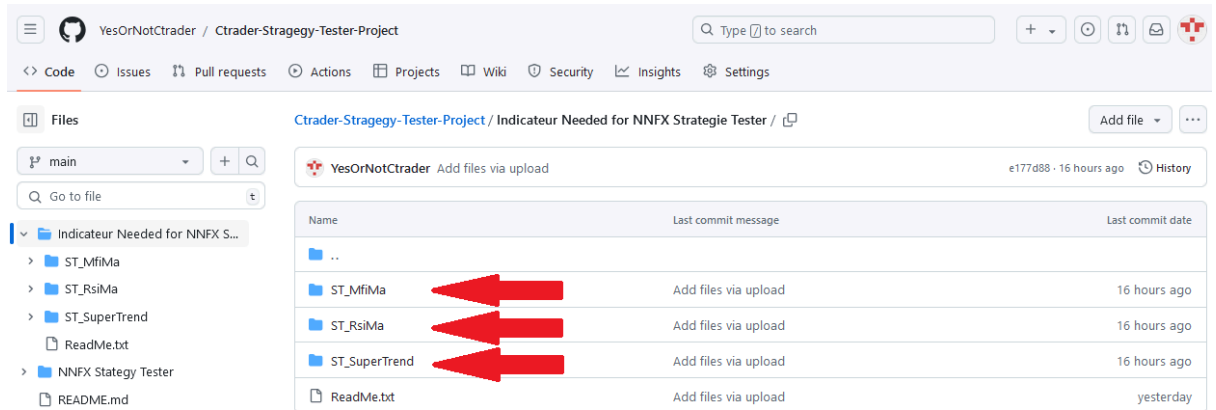
The README file is also visible, containing the following text:

Quantitative Analysis: The Key to Mastering Trading Strategies

Project to Develop a CBot for Trading Strategy Analysis

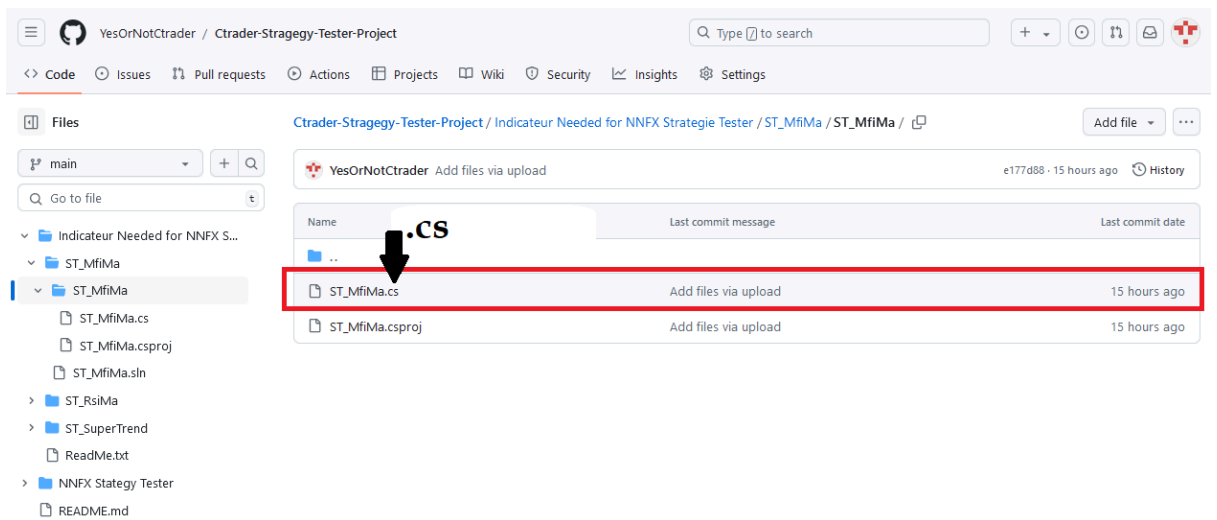
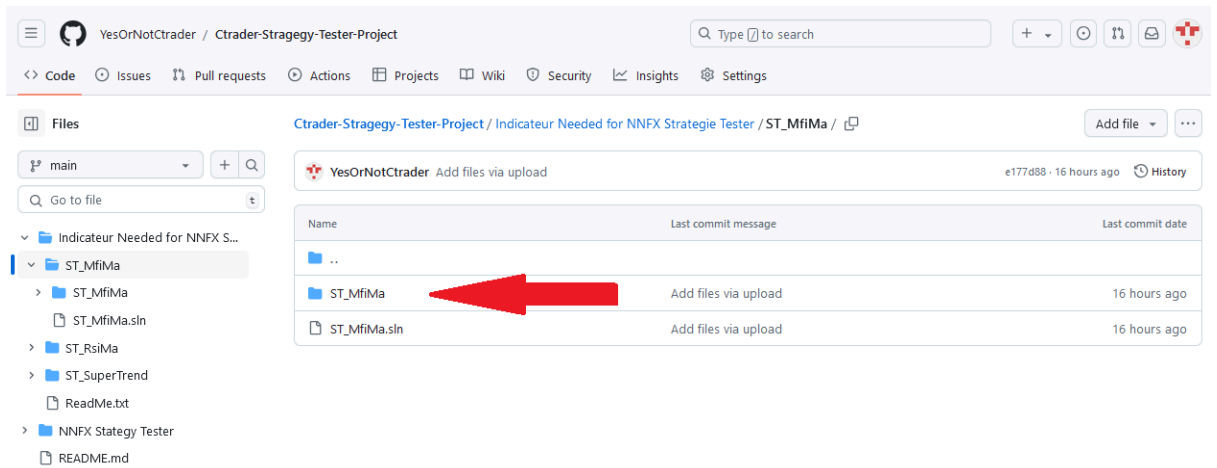
The objective of this project is to develop a CBot for the cTrader platform, capable of analyzing a wide range of trading strategies. This robot is based on the NNFX principle to identify effective trading strategies without relying on complex

3. Now, for each folder named "ST_...", click until you reach a file with the .cs extension.

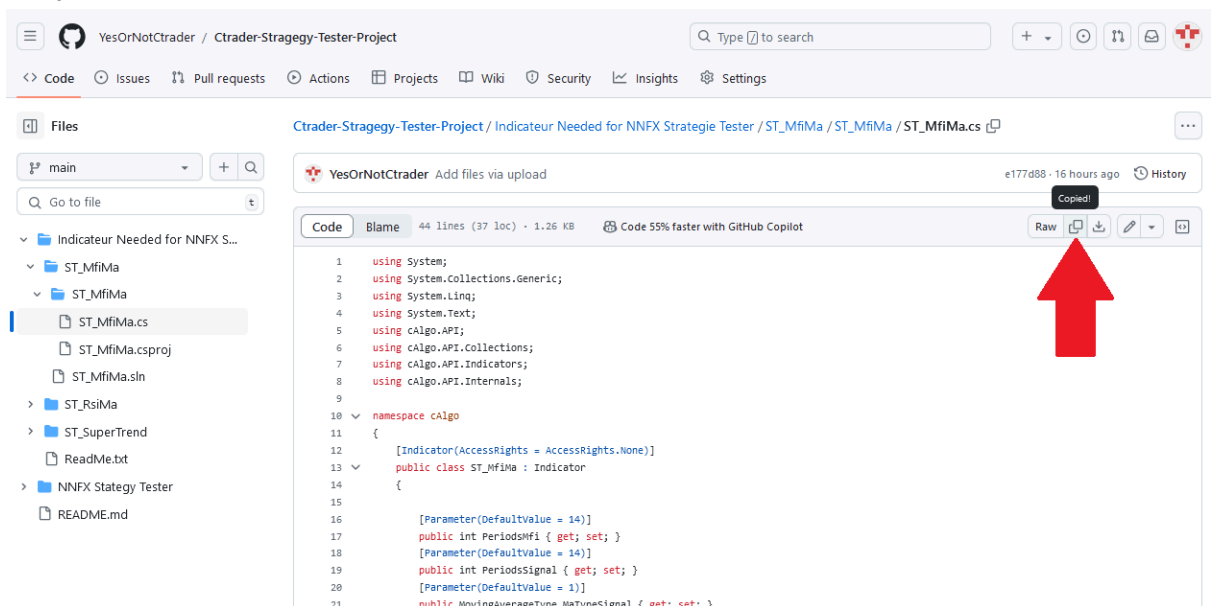


The screenshot shows the file structure of the repository. The left sidebar shows the file tree, and the main area shows the commit history for the 'Indicateur Needed for NNFX Strategie Tester' folder. Red arrows point to the folders 'ST_MfiMa', 'ST_RsiMa', and 'ST_SuperTrend'.

| Name | Last commit message | Last commit date |
|---------------|----------------------|------------------|
| .. | | |
| ST_MfiMa | Add files via upload | 16 hours ago |
| ST_RsiMa | Add files via upload | 16 hours ago |
| ST_SuperTrend | Add files via upload | 16 hours ago |
| ReadMe.txt | Add files via upload | yesterday |

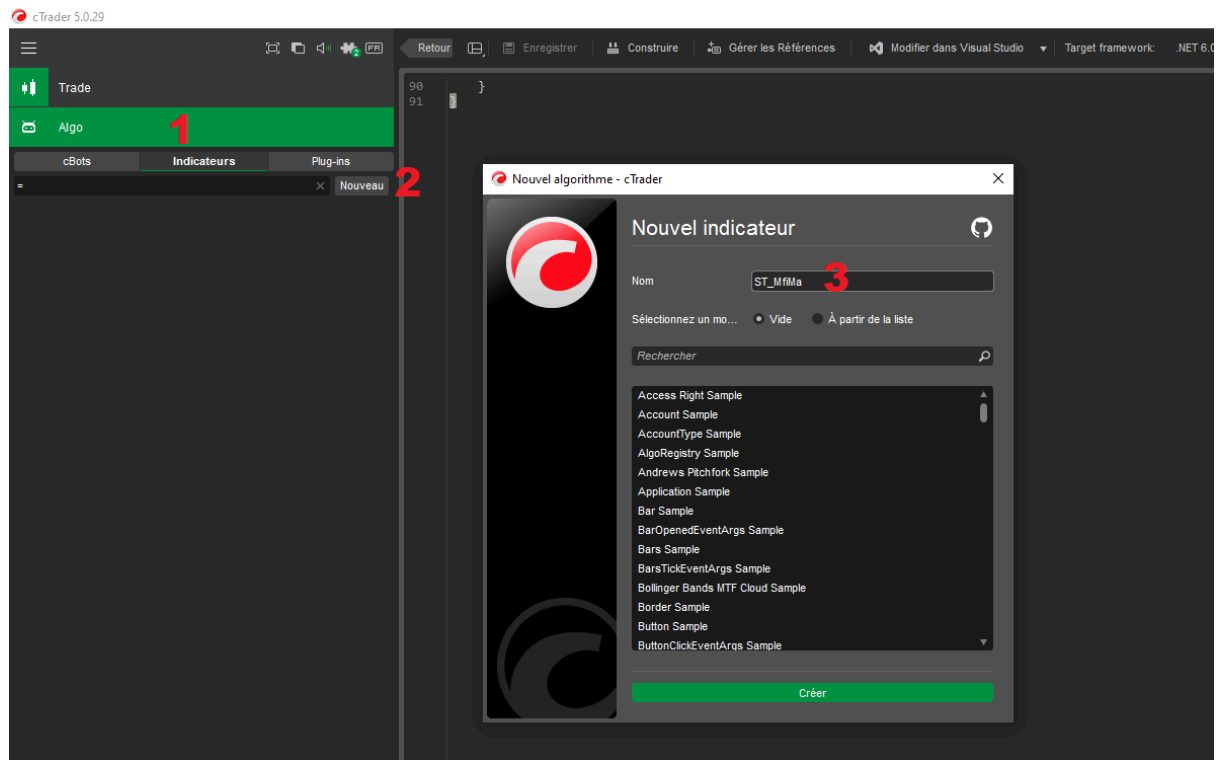


4. Copy the code obtained



- Go to cTrader, navigate to the Algo section, click on Indicators, and then on New. Name the indicator (we recommend using the name listed in the folder), then click

Create.



6. Once the indicator is created, do the following:
- Click on the code

```
1 using System;
2 using cAlgo.API;
3 using cAlgo.API.Collections;
4 using cAlgo.API.Indicators;
5 using cAlgo.API.Internals;
6
7 namespace cAlgo
8 {
9     [Indicator(AccessRights = AccessRights.None)]
10    public class ST_MfiMa : Indicator
11    {
12        [Parameter(DefaultValue = "Hello world!")]
13        public string Message { get; set; }
14
15        [Output("Main")]
16        public IndicatorDataSeries Result { get; set; }
17
18        protected override void Initialize()
19        {
20            // To learn more about cTrader Automate visit our Help Center:
21            // https://help.ctrader.com/ctrader-automate
22
23            Print(Message);
24        }
25
26        public override void Calculate(int index)
27        {
28            // Calculate value at specified index
29            // Result[index] =
30        }
31    }
32 }
```

- Press **Ctrl + A** (to select all)

```
1 using System;
2 using cAlgo.API;
3 using cAlgo.API.Collections;
4 using cAlgo.API.Indicators;
5 using cAlgo.API.Internals;
6
7 namespace cAlgo
8 {
9     [Indicator(AccessRights = AccessRights.None)]
10    public class ST_MfiMa1 : Indicator
11    {
12        [Parameter(DefaultValue = "Hello world!")]
13        public string Message { get; set; }
14
15        [Output("Main")]
16        public IndicatorDataSeries Result { get; set; }
17
18        protected override void Initialize()
19        {
20            // To learn more about cTrader Automate visit our Help Center:
21            // https://help.citrader.com/citrader-automate
22
23            Print(Message);
24        }
25
26        public override void Calculate(int index)
27        {
28            // Calculate value at specified index
29            // Result[index] =
30        }
31    }
32 }
```

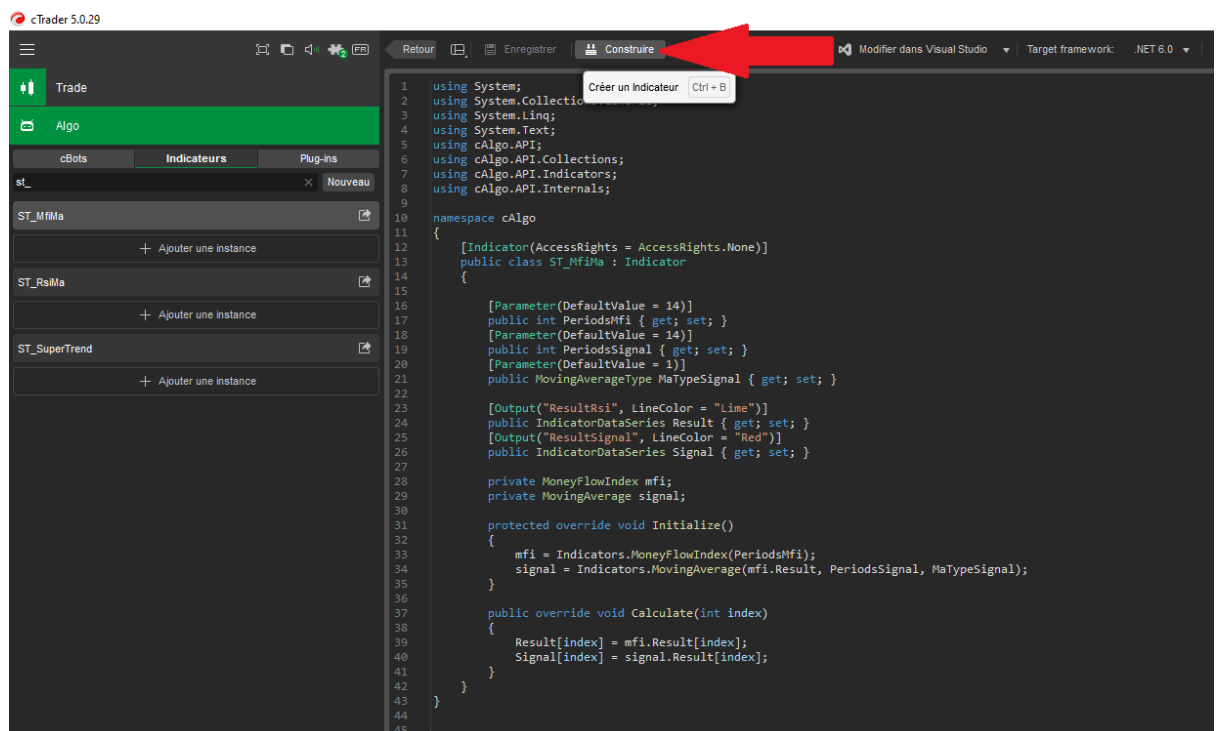
- Press **Ctrl + V** (to paste)

```

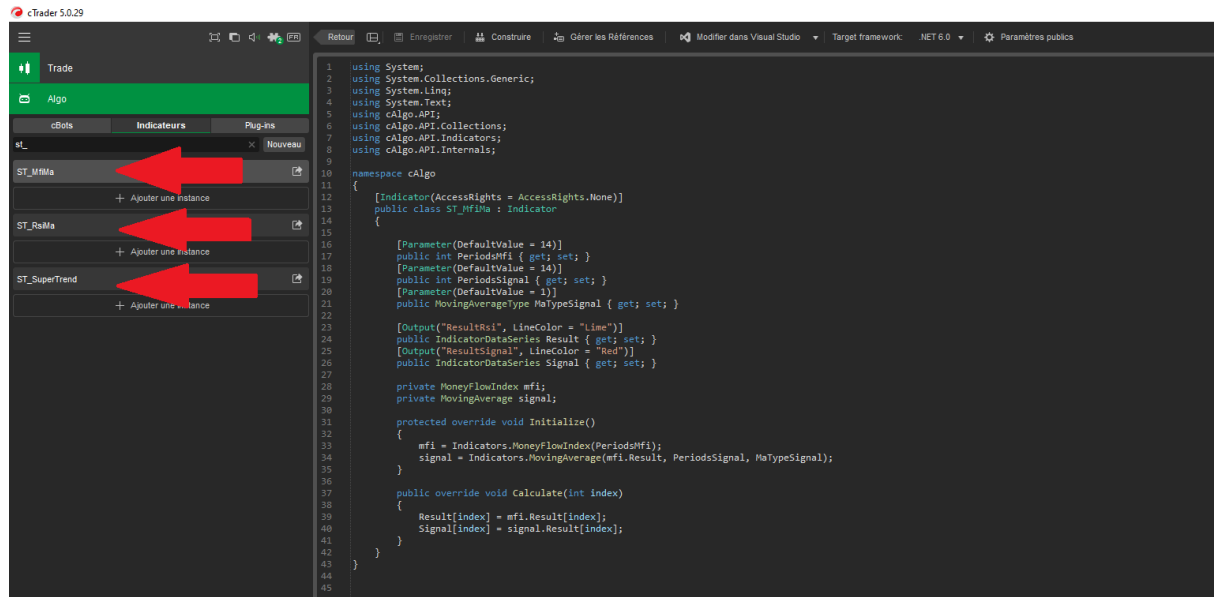
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using cAlgo.API;
6  using cAlgo.API.Collections;
7  using cAlgo.API.Indicators;
8  using cAlgo.API.Internals;
9
10 namespace cAlgo
11 {
12     [Indicator(AccessRights = AccessRights.None)]
13     public class ST_MfiMa : Indicator
14     {
15
16         [Parameter(DefaultValue = 14)]
17         public int PeriodsMfi { get; set; }
18         [Parameter(DefaultValue = 14)]
19         public int PeriodsSignal { get; set; }
20         [Parameter(DefaultValue = 1)]
21         public MovingAverageType MaTypeSignal { get; set; }
22
23         [Output("ResultRsi", LineColor = "Lime")]
24         public IndicatorDataSeries Result { get; set; }
25         [Output("ResultSignal", LineColor = "Red")]
26         public IndicatorDataSeries Signal { get; set; }
27
28         private MoneyFlowIndex mfi;
29         private MovingAverage signal;
30
31         protected override void Initialize()
32         {
33             mfi = Indicators.MoneyFlowIndex(PeriodsMfi);
34             signal = Indicators.MovingAverage(mfi.Result, PeriodsSignal, MaTypeSignal);
35         }
36
37         public override void Calculate(int index)
38         {
39             Result[index] = mfi.Result[index];
40             Signal[index] = signal.Result[index];
41         }
42     }
43 }
44
45

```

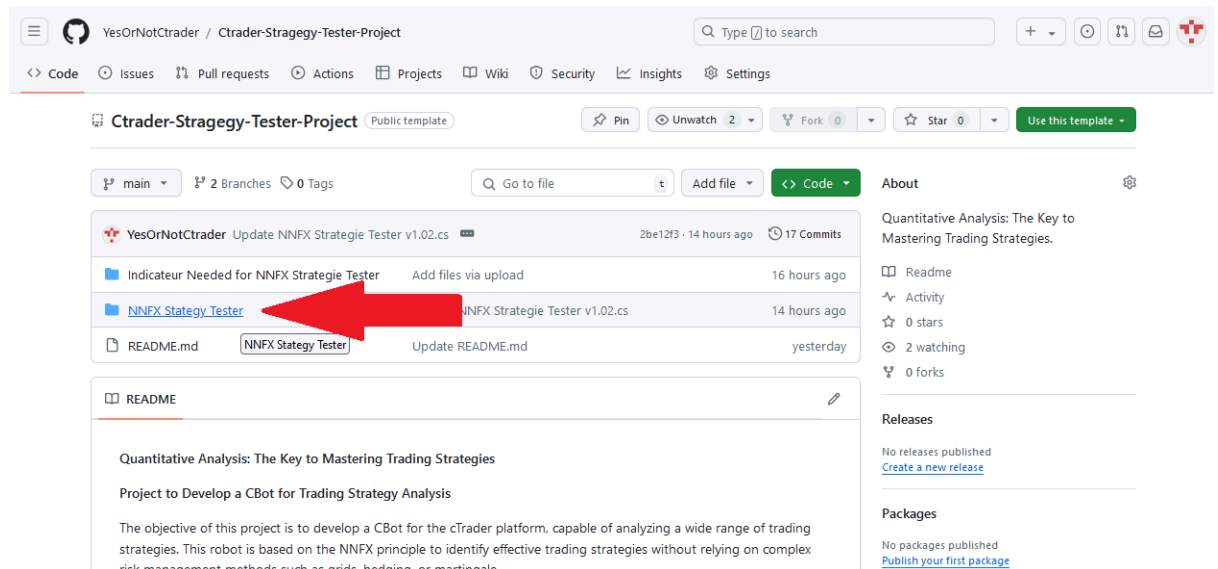
7. Finally, click Create. If you encounter an error, it's possible that the old indicator code was not cleared before you pasted the new code, so restart from step 4 and go directly to step 6.



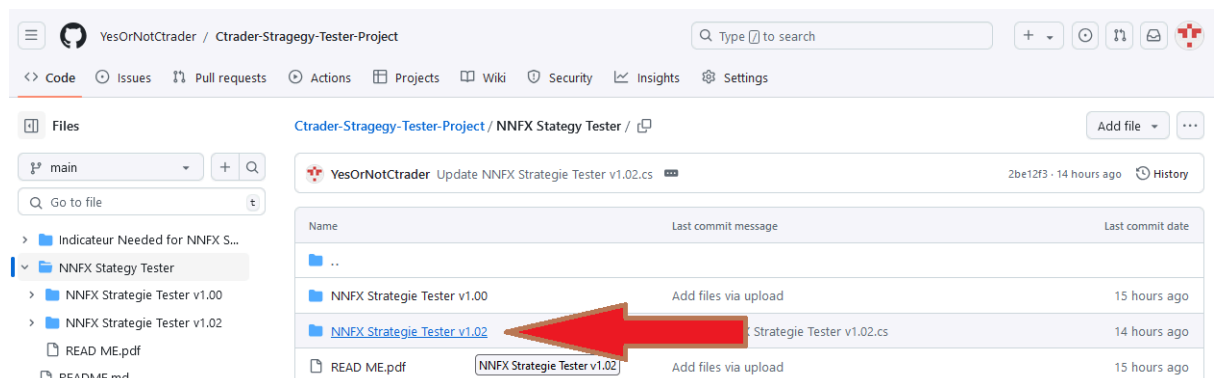
8. Repeat this process for all indicators named "ST_..."

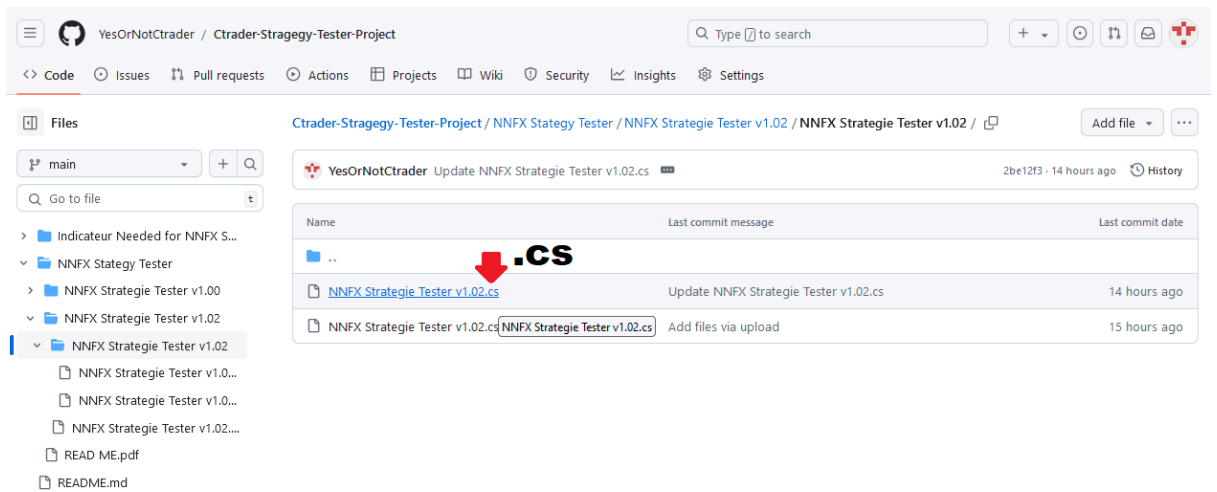
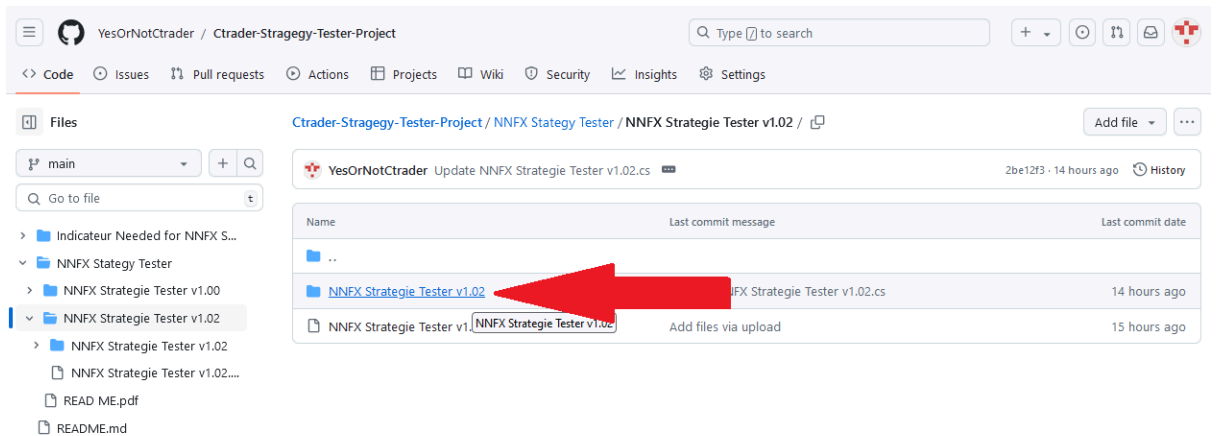


9. Once the indicators are created, click on "NNFX Strategy Tester."

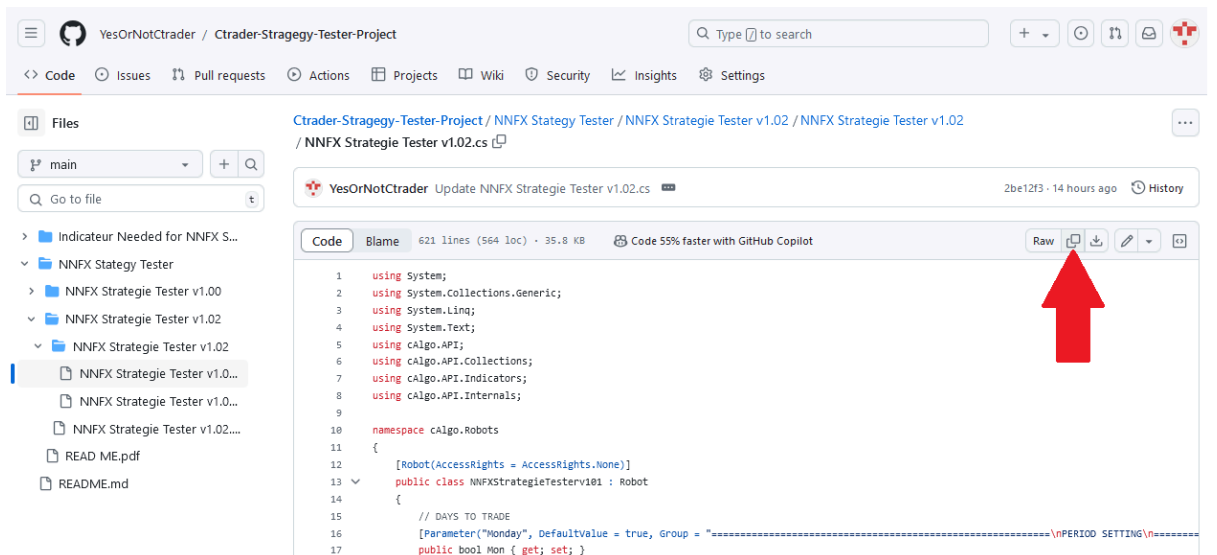


10. Now, select the latest version of "NNFX Strategy Tester v...". Click until you reach a file with the .cs extension.





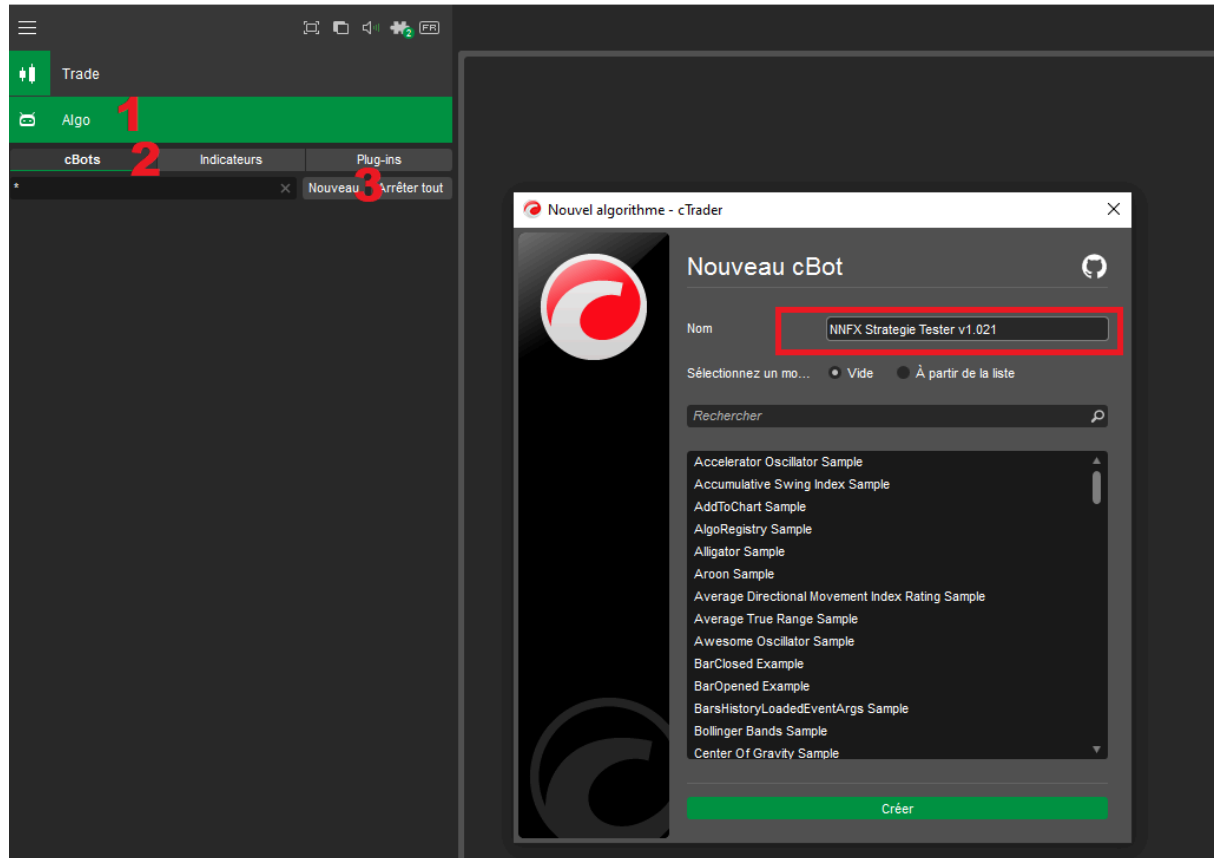
11. Copy the code obtained,



12. then go to cTrader, navigate to the Algo section, click on cBots, and then on New, Name the cBot (we recommend using the name listed in the folder), then click

Create.

cTrader 5.0.29



13. Once the cBot is created, do the following:

- Click on code

```
1  using System;
2  using cAlgo.API;
3  using cAlgo.API.Collections;
4  using cAlgo.API.Indicators;
5  using cAlgo.API.Internals;
6
7  namespace cAlgo.Robots
8  {
9      [Robot(AccessRights = AccessRights.None, AddIndicators = true)]
10     public class NAFXStrategieTesterv1021 : Robot
11     {
12         [Parameter(DefaultValue = "Hello world!")]
13         public string Message { get; set; }
14
15         protected override void OnStart()
16         {
17             // To learn more about cTrader Automate visit our Help Center:
18             // https://help.ctrader.com/ctrader-automate
19
20             Print(Message);
21         }
22
23         protected override void OnTick()
24         {
25             // Handle price updates here
26         }
27
28         protected override void OnStop()
29         {
30             // Handle cBot stop here
31         }
32     }
33 }
```

- Press **Ctrl + A** (to select all)

```
1  using System;
2  using cAlgo.API;
3  using cAlgo.API.Collections;
4  using cAlgo.API.Indicators;
5  using cAlgo.API.Internals;
6
7  namespace cAlgo.Robots
8  {
9      [Robot(AccessRights = AccessRights.None, AddIndicators = true)]
10     public class NAFXStrategieTesterv1021 : Robot
11     {
12         [Parameter(DefaultValue = "Hello world!")]
13         public string Message { get; set; }
14
15         protected override void OnStart()
16         {
17             // To learn more about cTrader Automate visit our Help Center:
18             // https://help.ctrader.com/ctrader-automate
19
20             Print(Message);
21         }
22
23         protected override void OnTick()
24         {
25             // Handle price updates here
26         }
27
28         protected override void OnStop()
29         {
30             // Handle cBot stop here
31         }
32     }
33 }
```

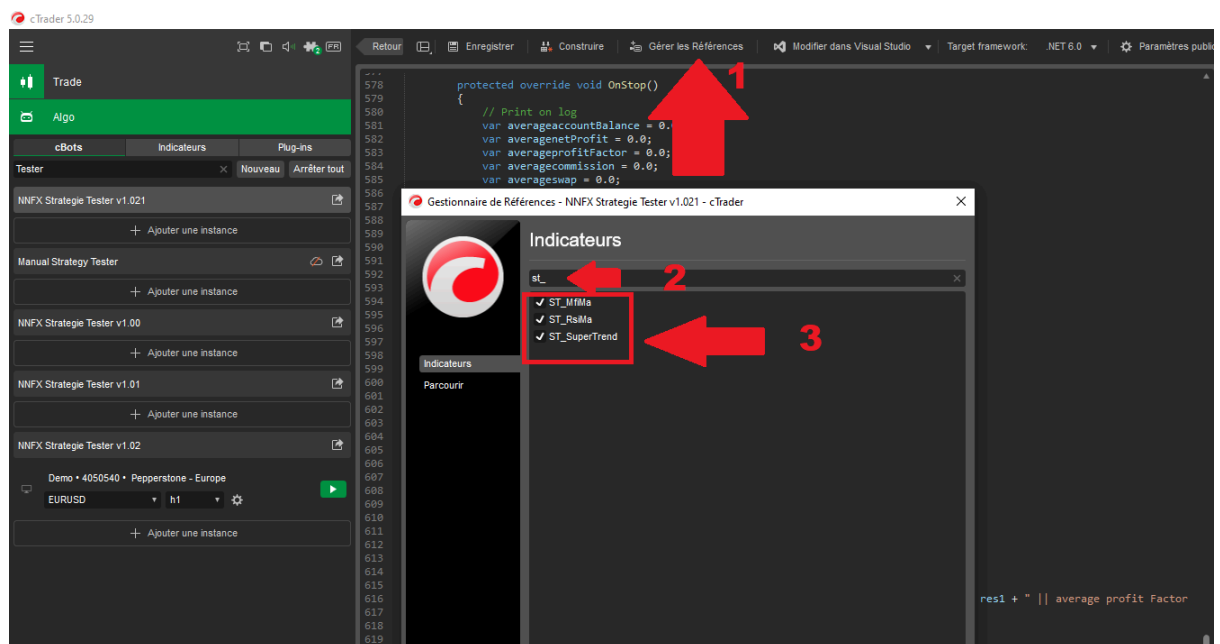
- Press **Ctrl + V** (to paste)

```

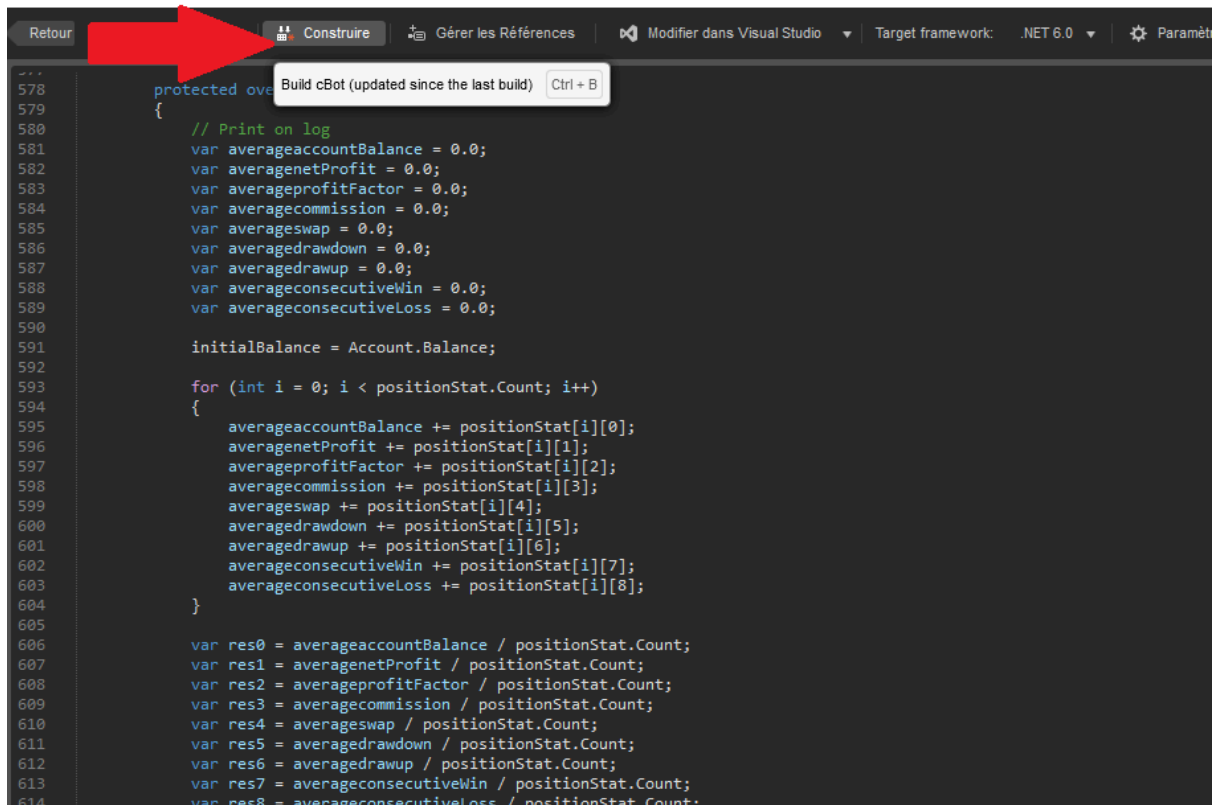
578     protected override void OnStop()
579     {
580         // Print on log
581         var averageaccountBalance = 0.0;
582         var averagenetProfit = 0.0;
583         var averageprofitFactor = 0.0;
584         var averagecommission = 0.0;
585         var averageswap = 0.0;
586         var averagedrawdown = 0.0;
587         var averagedrawup = 0.0;
588         var averageconsecutiveWin = 0.0;
589         var averageconsecutiveLoss = 0.0;
590
591         initialBalance = Account.Balance;
592
593         for (int i = 0; i < positionStat.Count; i++)
594         {
595             averageaccountBalance += positionStat[i][0];
596             averagenetProfit += positionStat[i][1];
597             averageprofitFactor += positionStat[i][2];
598             averagecommission += positionStat[i][3];
599             averageswap += positionStat[i][4];
600             averagedrawdown += positionStat[i][5];
601             averagedrawup += positionStat[i][6];
602             averageconsecutiveWin += positionStat[i][7];
603             averageconsecutiveLoss += positionStat[i][8];
604         }
605
606         var res0 = averageaccountBalance / positionStat.Count;
607         var res1 = averagenetProfit / positionStat.Count;
608         var res2 = averageprofitFactor / positionStat.Count;
609         var res3 = averagecommission / positionStat.Count;
610         var res4 = averageswap / positionStat.Count;
611         var res5 = averagedrawdown / positionStat.Count;
612         var res6 = averagedrawup / positionStat.Count;
613         var res7 = averageconsecutiveWin / positionStat.Count;
614         var res8 = averageconsecutiveLoss / positionStat.Count;
615
616         Print("average account Balance : " + res0 + " || average net Profit : " + res1 + " || average profit Factor
617
618         // Export to csv need Function need idea for good using
619     }
620 }
621

```

- Now that you have the code, click on **Manage References**. Select all the indicators necessary for this robot. Since we named them "ST_", enter "ST_" in the search bar and select all the listed indicators.



- Finally, click **Create**. If you encounter an error, it's possible that the old indicator code was not cleared before you pasted the new code, so restart from step 4 and go directly to step 6.



The screenshot shows a code editor window with a dark theme. At the top, there is a toolbar with several buttons: 'Retour', 'Construire' (highlighted with a red arrow), 'Gérer les Références', 'Modifier dans Visual Studio', 'Target framework: .NET 6.0', and 'Paramètres'. Below the toolbar, a tooltip for the 'Construire' button reads 'Build cBot (updated since the last build) Ctrl + B'. The code is in C# and is part of a class with a 'protected override' method. It includes a 'Print on log' comment and a series of variables for tracking account balance, profit, commission, swap, drawdown, and consecutive wins/losses. A 'for' loop iterates through 'positionStat' to calculate these values. Finally, it calculates eight results ('res0' through 'res8') by dividing the accumulated values by the count of 'positionStat'.

```
578     protected override void OnTick()
579     {
580         // Print on log
581         var averageaccountBalance = 0.0;
582         var averagenetProfit = 0.0;
583         var averageprofitFactor = 0.0;
584         var averagecommission = 0.0;
585         var averageswap = 0.0;
586         var averagedrawdown = 0.0;
587         var averagedrawup = 0.0;
588         var averageconsecutiveWin = 0.0;
589         var averageconsecutiveLoss = 0.0;
590
591         initialBalance = Account.Balance;
592
593         for (int i = 0; i < positionStat.Count; i++)
594         {
595             averageaccountBalance += positionStat[i][0];
596             averagenetProfit += positionStat[i][1];
597             averageprofitFactor += positionStat[i][2];
598             averagecommission += positionStat[i][3];
599             averageswap += positionStat[i][4];
600             averagedrawdown += positionStat[i][5];
601             averagedrawup += positionStat[i][6];
602             averageconsecutiveWin += positionStat[i][7];
603             averageconsecutiveLoss += positionStat[i][8];
604         }
605
606         var res0 = averageaccountBalance / positionStat.Count;
607         var res1 = averagenetProfit / positionStat.Count;
608         var res2 = averageprofitFactor / positionStat.Count;
609         var res3 = averagecommission / positionStat.Count;
610         var res4 = averageswap / positionStat.Count;
611         var res5 = averagedrawdown / positionStat.Count;
612         var res6 = averagedrawup / positionStat.Count;
613         var res7 = averageconsecutiveWin / positionStat.Count;
614         var res8 = averageconsecutiveLoss / positionStat.Count;
```

Congratulations, you did it!