



EventStoreDB From Scratch: Running Python in GitHub Codespaces

Overview

Welcome to the Python example of Event Store's **From Scratch** series. This series lets you quickly overcome the common challenges of setting up and configuring a new development environment and focus on advancing your EventStoreDB skills.

The **From Scratch** series provides working code examples for basic reads and writes to EventStoreDB, a tested environment to run the code, and instructions that clearly describe the steps required to run the code successfully.

Each **From Scratch** repository provides the following:

- A working GitHub Codespaces environment
- Instructions on running EventStoreDB locally
- Instructions to set up a similar project on your own

We recommend you progress through the **From Scratch** projects in the following order:

1. Run the code in Codespaces
2. Clone the **From Scratch** GitHub repo, and follow the instructions to run it locally
3. Build your own project

This document provides detailed instructions on launching GitHub Codespaces, starting an EventStoreDB Docker container, and running the sample Python code that writes and reads from EventStoreDB. ***This is the recommended starting point for running Python code with EventStoreDB.***

Other clients in the **From Scratch** series include:

- Node
- Java
- .Net

Topics covered

1. Launching Codespaces
2. Running code in Codespaces

1. Launching Codespaces

GitHub created Codespaces to answer the *"It doesn't run on my machine"* problem faced by developers of all experience levels. Codespaces provides the IDE, the repo, and the environment so you can focus on running code.

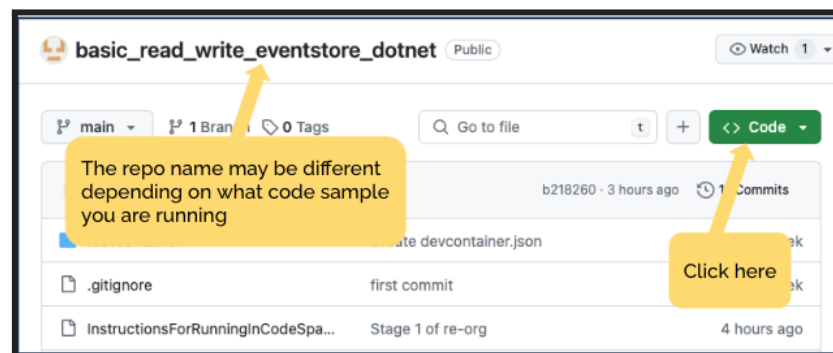
Here are the steps to launch the **FromScratch** repos in GitHub Codespaces.

Requirements

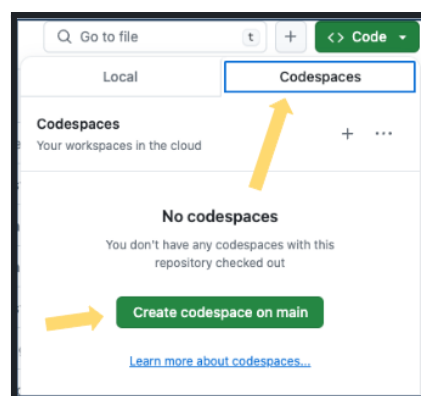
1. A GitHub account
2. A browser
3. Access to the internet

Steps

1. Navigate to the [Python FromScratch repository](#).
2. Click on the green "<> Code" button



3. You will see two tabs titled "Local" and "Codespaces." Select "Codespaces," then click the green button labeled "Create codespace on main."



4. Wait for your Codespace to launch. Depending on the container's configurations, this can take anywhere from a few seconds to a few minutes. While it launches you will see this image.



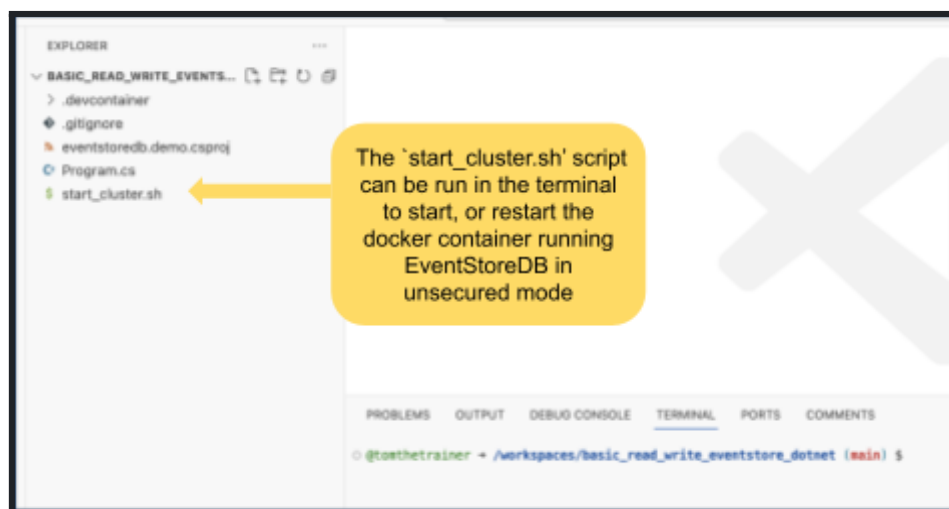
1. You can make some formatting choices on the welcome page of VS Code (embedded in Codespaces). VS Code is the default IDE in Codespaces for the **From Scratch** project. Choose your preferred theme or use the default theme by closing the "Welcome" tab.

2. Running the Code in Codespaces

You'll need a running cluster to read and write code with EventStoreDB.

Using a Docker container is a quick way to get started. More information is available at <https://developers.eventstore.com/server/v24.2/installation.html#docker>

We provide a shell script for the **From Scratch** project that starts or restarts a Docker container running EventStoreDB.



Some notes on the 'start_cluster.sh'

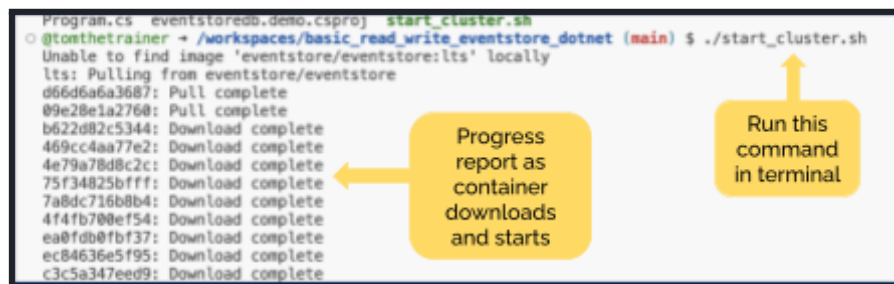
The 'start_cluster.sh' script is designed to either start, or in the case of an already running Docker container, to restart the container. Restarting the container with **start_cluster.sh** will delete any streams you had written to the previous instance of the Docker container. This design decision is intentional.

Please note that Codespaces are set to pause after a period of inactivity. When restarting an inactivated Codespace the **start_cluster.sh** script may fail to restart the Docker container. The most straightforward solution to this issue is terminating the Codespace and starting a new one.

Follow the steps below to start your cluster.

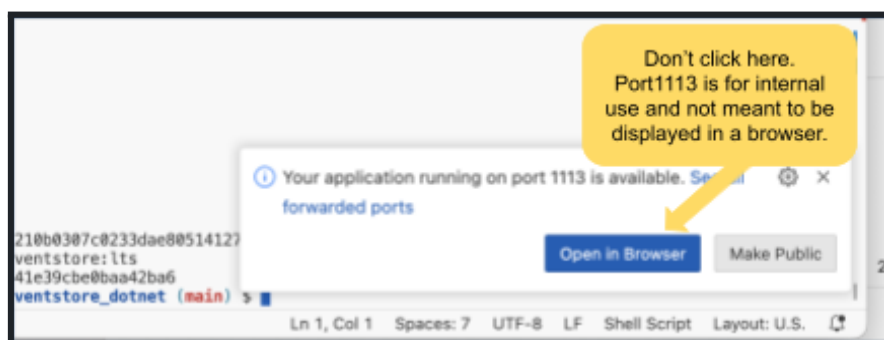
1. To launch a Docker container running EventStoreDB where the "FromScratch" code will write and read events, run the **start_cluster.sh** script. Type the following command into the terminal located at the bottom of your Codespace.

```
./start_cluster.sh
```



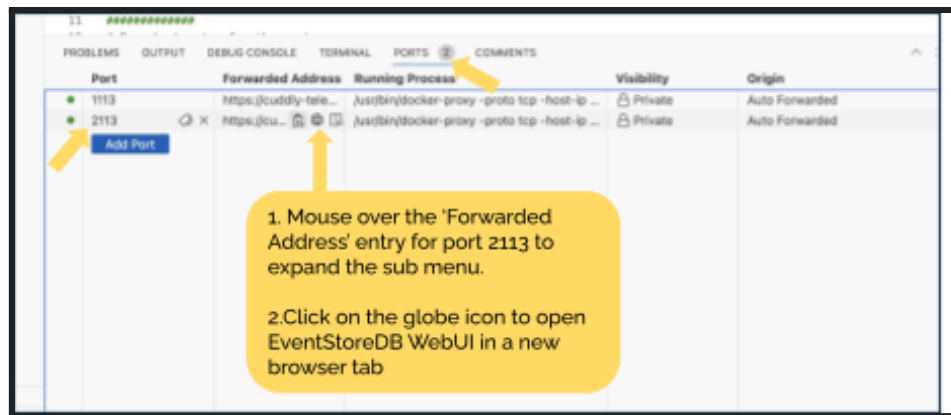
```
Program.cs eventstoredb.demo.csproj start_cluster.sh
@tomthetrainer → /workspaces/basic_read_write_eventstore_dotnet (main) $ ./start_cluster.sh
Unable to find image 'eventstore/eventstore:ltts' locally
ltts: Pulling from eventstore/eventstore
d66d6a6a3687: Pull complete
09e28e1a2760: Pull complete
b622d82c5344: Download complete
469cc4aa77e2: Download complete
4e79a78d8c2c: Download complete
75f34825bfff: Download complete
7a8dc716b8b4: Download complete
4f4fb700ef54: Download complete
ea0fdb0fbf37: Download complete
ec84636e5f95: Download complete
c3c5a347eed9: Download complete
```

Once your Docker container has finished downloading, you may see a pop-up stating, "Your application running on port 1113 is available..." **Do not click** "Open in Browser." Port :1113 is used for RPC calls and will not direct you to the WebUI (which you will do in the next step).

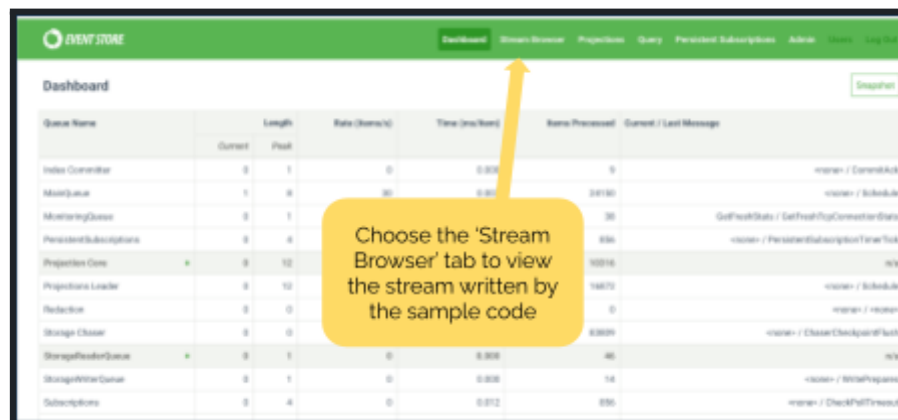


Access the EventStoreDB WebUI Stream Browser

1. Open the WebUI of EventStoreDB running in the Docker container. EventStoreDB uses ports :1113 and :2113. Open the WebUI '**port :2113**' in a browser tab.



2. Select the Stream Browser tab from the EventStoreDB WebUI. After running the sample append code, the events written in the demo will be visible here.

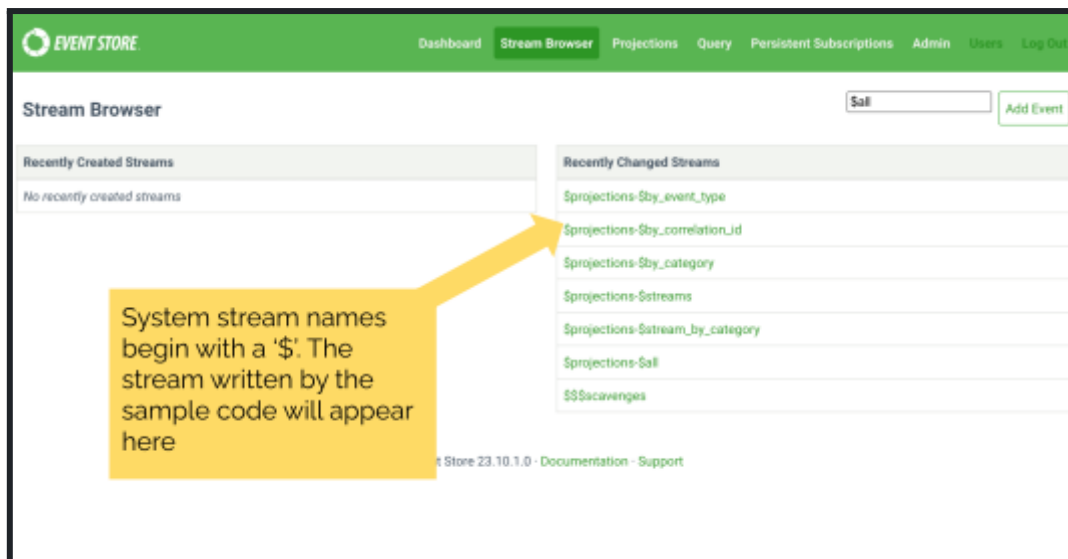


Stream Browser view explained

The 'Stream Browser' tab provides an overview of recently created and changed streams. Clicking on a stream name shows details about the individual stream.

A system stream in EventStoreDB is a type of stream that is used for system-level operations and information. A "\$" prefix distinguishes these streams. For example, the stream metadata for a stream named "foo" is \$foo. System streams can contain metadata for other streams or system-level information.

You can ignore system streams for this example. However, as you continue on your EventStoreDB journey, you will find them useful information sources.



Congratulations! You have successfully started the EventStoreDB cluster and viewed the stream browser from the EventStoreDB WebUI.

Run the Python code sample

1. View the WebUI to verify that you have a running EventStoreDB cluster. If you need to start the cluster, run the following in a terminal window.

```
./start_cluster.sh
```

2. Open the stream browser in the WebUI of the cluster
3. In the VS Code Explorer tab on the left, you will see:

.gitignore	(file exclusion list for GitHub)
CodeSpacesInstructions.pdf	(Instructions for CodeSpaces)
InstructionsForRunningLocally.pdf	(Instructions for running locally)
README.md	(The README file for the repo)
requirements.txt	(A python dependency management file)
sample_append.py	(a python program to append an event)
sample_read.py	(A python program to read events)
start_cluster.sh	(a shell script that starts EventStoreDB docker)

4. Execute the program sample_append.py by typing the following in the terminal window at the bottom of the screen.

```
python sample_append.py
```

5. Verify that an event has been written to EventStoreDB by viewing the Stream Browser on the EventStoreDB WebUI.
6. Execute the program `sample_read.py` by running the following command in the terminal. You should see the event previously written displayed on the console.

```
python sample_read.py
```

Congratulations! After running `sample_append.py` followed by `sample_read.py` you have succeeded in Writing and Reading events to and from EventstoreDB.

Next Steps

Now that you have successfully leveraged GitHub Codespaces to read and write code to EventStoreDB, we recommend you continue your learning with the **From Scratch** Python instructions for running code locally.

As you progress with your EventStoreDB skills, you can also find additional examples in the following repo:

<https://github.com/EventStore/samples>

In particular, we recommend the Quickstart examples here:

<https://github.com/EventStore/samples/tree/main/Quickstart>