EventStoreDB From Scratch: Running .NET in GitHub Codespaces

Overview

Welcome to the .NET (C#) example of Event Store's **From Scratch** series. This series allows you to 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 EnventStoreDB docker container, and running the sample .NET code that writes and reads from EventStoreDB. ***This is the recommended starting point for running .NET code with EventStoreDB.***

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

- Node
- Java
- Python

Topics covered

1. Launching Codespaces
2. Running code in Codespaces

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.

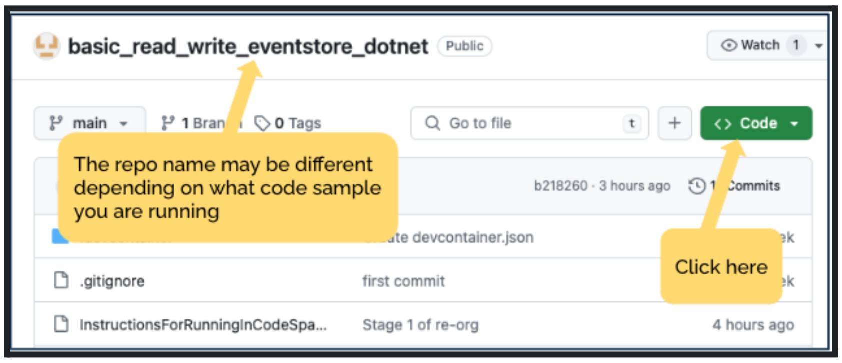
Requirements

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

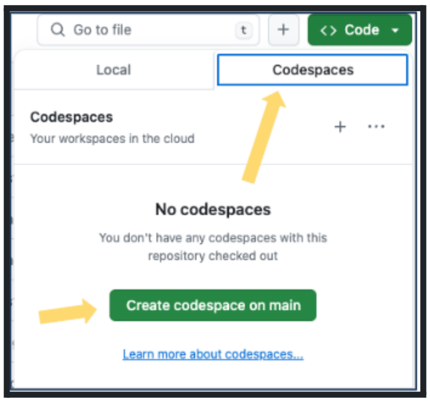
Steps

1. Navigate to the .NET **FromScratch** repository.

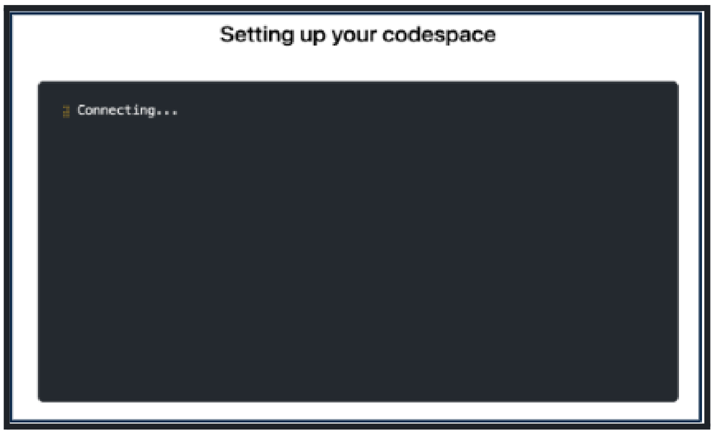
2. Click on the green "<> Code" button



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


Go to file
Code
Local
Codespaces
Codespaces
Your workspaces in the cloud
No codespaces
You don't have any codespaces with this repository checked out
Create codespace on main
Learn more about codespaces...

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.



```
Connecting...
```

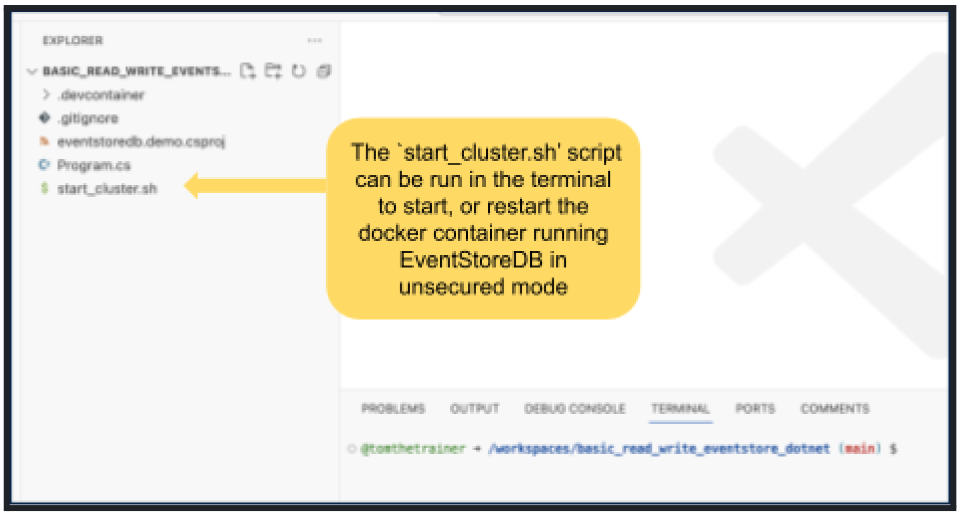
5. On the welcome page of VS Code, you will be given the option to make some formatting choices. VS Code is the default IDE in Codespaces for the **From Scratch** project. Choose your preferred theme or use the default theme by closing the “Welcome” tab.

Running the Code in Codespaces

To read and write code with EventstoreDB, you'll need a running cluster.

Using a docker container is a quick way to get started. More information is available here, https://developers.eventstore.com/server/v24.2/installation.html#run-with-docker

For the “From Scratch” project we provide a shell script that starts or restarts a docker container running EventStoreDB.


EXPLORER
BASIC_READ_WRITE_EVENTS...
.devcontainer
.gitignore
eventstoredb.demo.csproj
Program.cs
start_cluster.sh
The `start_cluster.sh' script can be run in the terminal to start, or restart the docker container running EventStoreDB in unsecured mode
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS

```
@tomthetrainer → /workspaces/basic_read_write_eventstore_dotnet (main) $
```

Some notes on the 'start_cluster.sh'

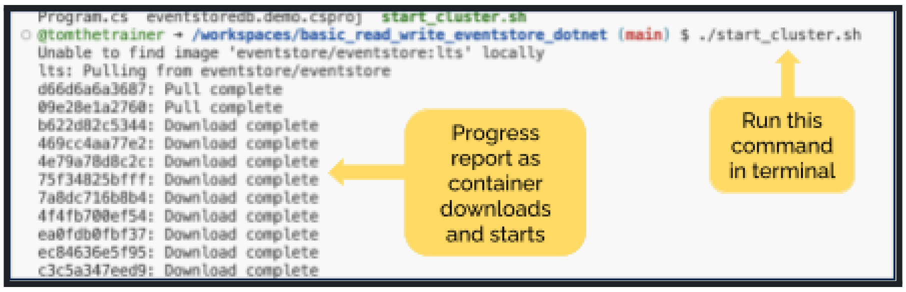
The 'start_cluster.sh' script is designed to either start, or in the case of an already running docker container, to restart the container. Restarting the container with '`start_cluster.sh`' will delete any streams that you had written to the previous instance of the docker container. This design decision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

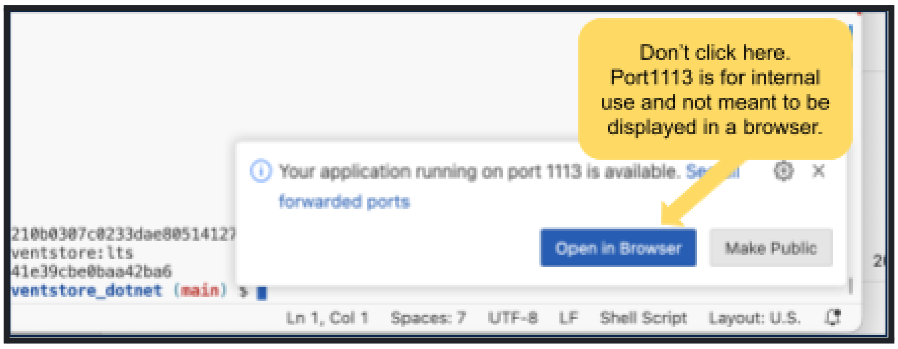
```
./start_cluster.sh
```



```
  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:lts' locally
  lts: 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
```

Progress report as container downloads and starts
Run this command in terminal

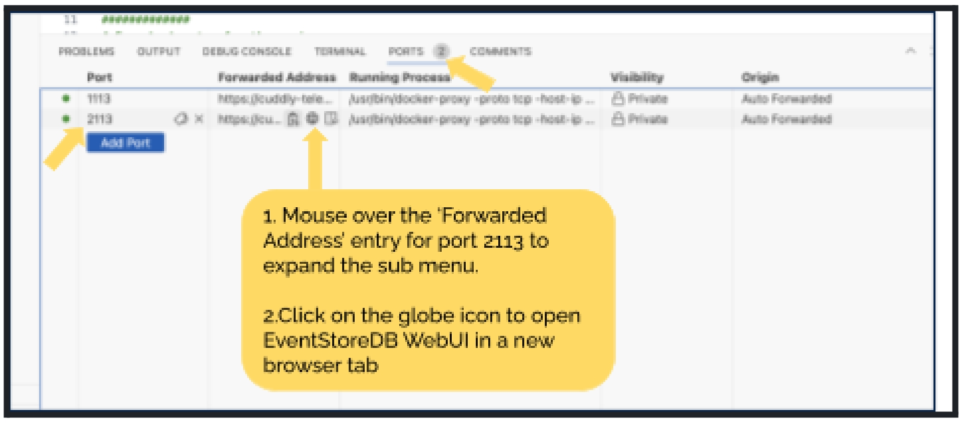
Once your Docker container has finished downloading, you may see a pop-up stating, "*Your application running on port 113 is available…*" Do not click "Open in Browser." Port :1113 is used for RPC calls and will not direct you to the WebUI (which you will do in the next step).


Don't click here. Port1113 is for internal use and not meant to be displayed in a browser.
Your application running on port 1113 is available. See all forwarded ports
Open in Browser
Make Public
Ln 1, Col 1 Spaces: 7 UTF-8 LF Shell Script Layout: U.S.

```
210b0307c0233dae80514127
ventstore:lts
41e39cbe0baa42ba6
ventstore_dotnet (main) $
```

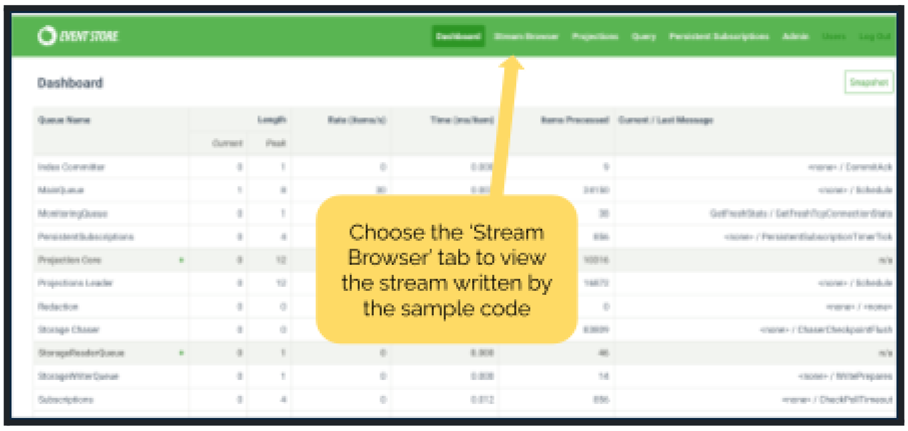
Access the EventStoreDB WebUI Stream Browser

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.


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS 2
COMMENTS
Add Port
1. Mouse over the 'Forwarded Address' entry for port 2113 to expand the sub menu.
2.Click on the globe icon to open EventStoreDB WebUI in a new browser tab

Port	Forwarded Address	Running Process	Visibility	Origin
1113	https://cuddly-tele...	/usr/bin/docker-proxy -proto tcp -host-ip ...	Private	Auto Forwarded
2113	https://cu...	/usr/bin/docker-proxy -proto tcp -host-ip ...	Private	Auto Forwarded

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.



Dashboard

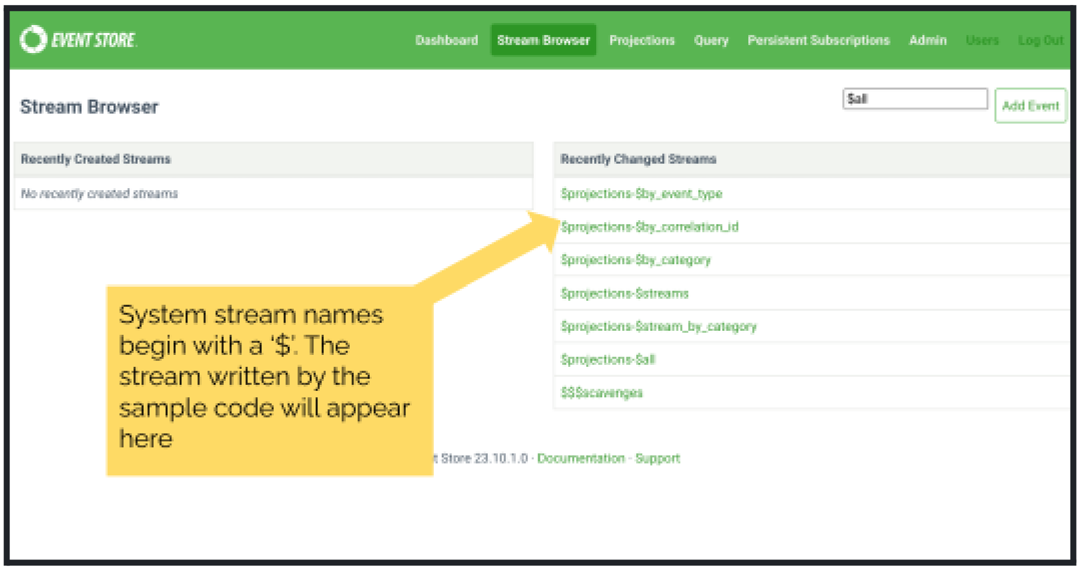
Queue Name	Length		Rate (items/s)	Time (ms/item)	Items Processed	Current / Last Message
	Current	Peak				
Index Committer	0	1	0	[illegible]	5	<none> / CommitAck
MainQueue	1	8	30	[illegible]	24130	<none> / Schedule
MonitoringQueue	0	1	[illegible]	[illegible]	30	GetFreshStats / GetFreshTcpConnectionStats
PersistentSubscriptions	0	4	[illegible]	[illegible]	656	<none> / PersistentSubscriptionTimerTick
Projection Core	0	12	[illegible]	[illegible]	10016	n/a
Projections Leader	0	12	[illegible]	[illegible]	14673	<none> / Schedule
Redaction	0	0	[illegible]	[illegible]	0	<none> / <none>
Storage Chaser	0	0	[illegible]	[illegible]	63609	<none> / ChaserCheckpointFlush
StorageReaderQueue	0	1	0	0.008	46	n/a
StorageWriterQueue	0	1	0	0.008	14	<none> / WritePrepares
Subscriptions	0	4	0	0.012	656	<none> / CheckPollTimeout

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. These streams are distinguished by a $ prefix. For example, the stream metadata for a stream named foo is $foo. System streams can contain metadata for other streams or system-level information.

For the purposes of this example you can ignore system streams. As you continue on your EventStoreDB journey, you will find them to be useful sources of information.



Recently Created Streams
No recently created streams

Recently Changed Streams
$projections-$by_event_type
$projections-$by_correlation_id
$projections-$by_category
$projections-$streams
$projections-$stream_by_category
$projections-$all
$$$scavenges

System stream names begin with a '$'. The stream written by the sample code will appear here

t Store 23.10.1.0 · Documentation · Support

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

Run the .NET(C#) code sample

1. Verify you have a running EventStoreDB cluster by viewing the WebUI. 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:

.devcontainer	(a file to manage Codespaces)
.gitignore	(file exclusion list for GitHub)
event store db.demo.csproj	(configuration file for C#)
program.cs	(a program that writes events to EventStoreDB)
Start_cluster.sh	(a shell script that starts EventStoreDB docker)

4. Execute the program.cs file running “dotnet run” in the terminal window. This will add “bin” and “obj” folders to the current directory as the run command builds the needed resources to execute.

   Also note that program.cs reads the event back after writing it and displays the contents of the events in the stream in the terminal. Running multiple times will add events to the stream.

5. Verify that an event has been written to EventStoreDB by viewing the Stream Browser on the EventStoreDB WebUI.

Congratulations! After running program.cs 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 run code locally. Please continue to the **From Scratch** .NET instructions for running locally to continue your learning.

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