SEP 7, 2023 READ TIME: 6 MIN

Is Windows your favorite development environment? Do you want to run Apache Kafka® on Windows? Thanks to the Windows Subsystem for Linux 2 (WSL 2), now you can, and with fewer tears than in the past. Windows still isn’t the recommended platform for running Kafka with production workloads, but for trying out Kafka, it works just fine.

Let’s take a look at how it’s done:

- Set up your environment
  - Install Windows Subsystem for Linux
  - Enable WSL 2
  - Install Java

Table of Contents

Set up your environment

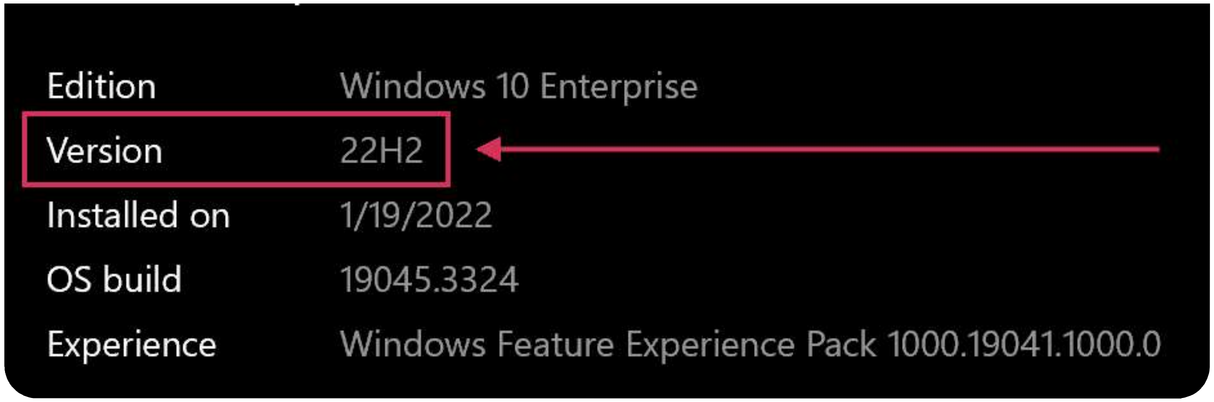
The Windows Subsystem for Linux (WSL 2) makes it all possible. Microsoft describes WSL as “a GNU/Linux environment—including most command line tools, utilities, and applications—directly on Windows, unmodified, without the overhead of a traditional virtual machine or dual boot setup.”

Install Windows Subsystem for Linux

If you already have WSL 2 installed, skip to Install Java.

Make sure you’re running Windows 10 version 21H1 or later, or Windows 11 21H2 or later. To check your version, navigate to **Settings > System > About**. In the **Windows specifications** section, find **Version**. Ensure that you have all updates to your version installed.

WRITTEN BY



Edition Windows 10 Enterprise
Version 22H2
Installed on 1/19/2022
OS build 19045.3324
Experience Windows Feature Experience Pack 1000.19041.1000.0

windows-version-22h2

If you’re on the Windows Update train, you probably have the latest version and are good to go. If not, you need to update Windows.

When you’re sure that Windows is up to date, follow these instructions to install WSL 2.

Enable WSL 2

To install the WSL 2 feature, open PowerShell as an administrator, and run the following command:

```
wsl --install
```

ay take a minute or two. Your output should resemble the following:

```
PS C:\Windows\system32> wsl --install
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Installing: Ubuntu
Ubuntu has been installed.
The requested operation is successful. Changes will not be effective until the sy
PS C:\Windows\system32>
```

This command installs the Microsoft Store version of WSL and automatically selects the WSL 2 version. Also, it installs the default Linux distro, which is Ubuntu 22.04 as of this writing.

Tip: If you already have the non-Store version of WSL installed, you can run the `wsl –update` command to get it.

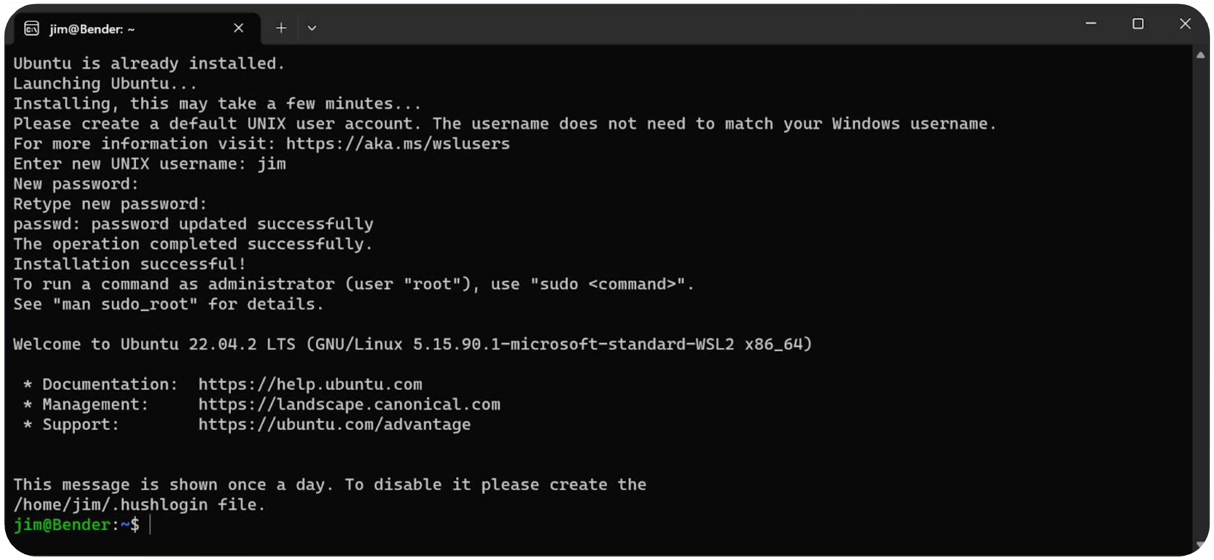
Reboot your machine.

After the reboot completes, log in. The installation of the default Linux distro continues automatically. The shell terminal opens and displays the following message:

Copy

```
Installing, this may take a few minutes...
  ease create a default UNIX user account. The username does not need to match yo
```

Enter a username and password to complete the installation. Save them in a secure location, because you will need them when you work with the shell later.



```
Ubuntu is already installed.
Launching Ubuntu...
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: jim
New password:
Retype new password:
passwd: password updated successfully
The operation completed successfully.
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


This message is shown once a day. To disable it please create the
/home/jim/.hushlogin file.
jim@Bender:~$
```

WSL 2 is ready to use. For more information on installing WSL 2, including troubleshooting, see Install Linux on Windows with WSL. For more information on WSL commands, see Basic commands for WSL.

Install Java

is built with Java and requires the Java runtime to execute. You can use t-get package manager to install the latest updates. In the Ubuntu shell

```
sudo apt-get update && sudo apt-get upgrade -y
```

Tip: Right-click pastes text into the terminal window.

It may take a few minutes to download and install all of the most recent binaries. Once Ubuntu is updated, you can install Java.

Kafka requires the Java runtime version to be 8, 11, or 17. Java 8 is deprecated, so Java 11 and Java 17 are preferred. Check the Java version in your Linux installation:

```
java -version
```

Your output should resemble this:

```
openjdk 11.0.17 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu220.04)
```

If Java isn’t installed (likely) or it’s not the right version, install it by using your distribution’s package manager. There are a lot of ways to install Java. On Ubuntu, this is one of the simplest:

Copy

```
sudo apt install openjdk-17-jdk -y
```

Download Kafka

You can install Kafka by using a package manager, or you can download the tarball and extract it to your local machine directly.

Download the tarball from the Apache Kafka download site. The following command downloads Apache Kafka version 3.5:

Copy

```
wget https://dlcdn.apache.org/kafka/3.5.0/kafka_2.13-3.5.0.tgz
```

Run the following commands to untar the Kafka archive, and `cd` to the `kafka` ory:

```
tar -xzf kafka_2.13-3.5.0.tgz
cd kafka_2.13-3.5.0
```

Run the `ls -al` command to list the contents of the `kafka` directory:

```
total 72
drwxr-xr-x  7 jim jim  4096 Jun  5 02:08 .
drwxr-xr-x 59 jim jim  4096 Aug 16 14:09 ..
-rw-r--r--  1 jim jim 14770 Jun  5 02:03 LICENSE
-rw-r--r--  1 jim jim 28184 Jun  5 02:03 NOTICE
drwxr-xr-x  3 jim jim  4096 Jun  5 02:08 bin
drwxr-xr-x  3 jim jim  4096 Jun  5 02:08 config
drwxr-xr-x  2 jim jim  4096 Aug 16 14:09 libs
drwxr-xr-x  2 jim jim  4096 Jun  5 02:08 licenses
drwxr-xr-x  2 jim jim  4096 Jun  5 02:08 site-docs
```

Start the Kafka cluster

Run the `kafka-storage.sh` script to generate a cluster ID:

```
FKA_CLUSTER_ID="$(bin/kafka-storage.sh random-uuid)"
```

```
bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c config/kraft/server.propertie
```

Your output should resemble:

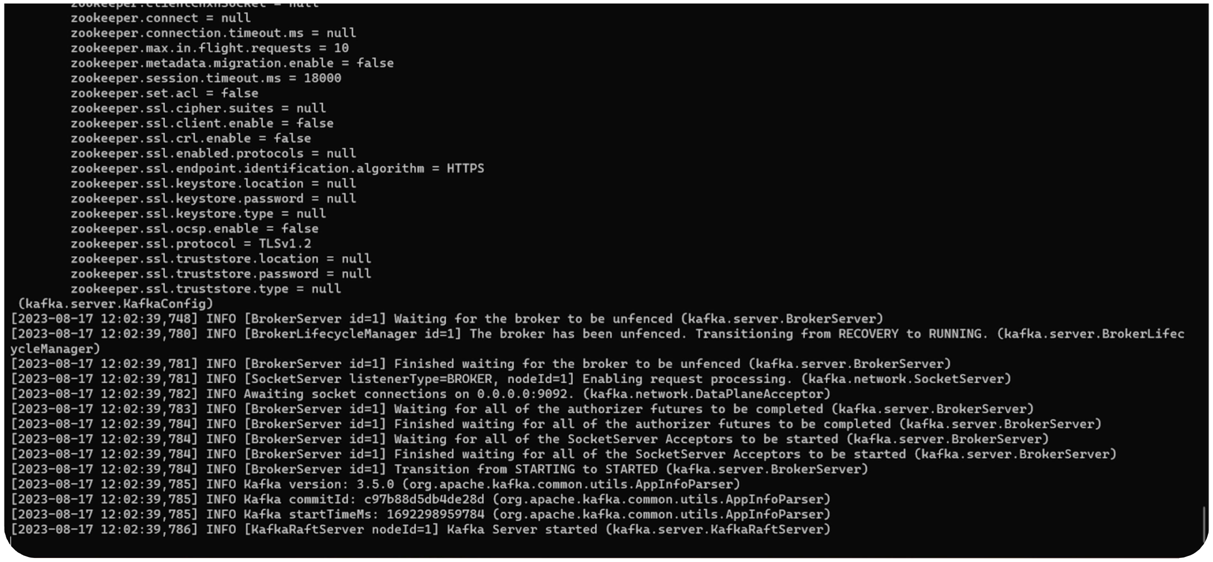
```
Formatting /tmp/kraft-combined-logs with metadata.version 3.5-IV2.
```

Run the `kafka-server-start.sh` script to start the Kafka server:

```
bin/kafka-server-start.sh config/kraft/server.properties
```

There will be a lot of output, and Kafka Server will be ready in a short time, typically around a second or two.

Your output should resemble the following screenshot:



```
        zookeeper.clientCnxnSocket = null
        zookeeper.connect = null
        zookeeper.connection.timeout.ms = null
        zookeeper.max.in.flight.requests = 10
        zookeeper.metadata.migration.enable = false
        zookeeper.session.timeout.ms = 18000
        zookeeper.set.acl = false
        zookeeper.ssl.cipher.suites = null
        zookeeper.ssl.client.enable = false
        zookeeper.ssl.crl.enable = false
        zookeeper.ssl.enabled.protocols = null
        zookeeper.ssl.endpoint.identification.algorithm = HTTPS
        zookeeper.ssl.keystore.location = null
        zookeeper.ssl.keystore.password = null
        zookeeper.ssl.keystore.type = null
        zookeeper.ssl.ocsp.enable = false
        zookeeper.ssl.protocol = TLSv1.2
        zookeeper.ssl.truststore.location = null
        zookeeper.ssl.truststore.password = null
        zookeeper.ssl.truststore.type = null
 (kafka.server.KafkaConfig)
[2023-08-17 12:02:39,748] INFO [BrokerServer id=1] Waiting for the broker to be unfenced (kafka.server.BrokerServer)
[2023-08-17 12:02:39,780] INFO [BrokerLifecycleManager id=1] The broker has been unfenced. Transitioning from RECOVERY to RUNNING. (kafka.server.BrokerLifec
ycleManager)
[2023-08-17 12:02:39,781] INFO [BrokerServer id=1] Finished waiting for the broker to be unfenced (kafka.server.BrokerServer)
[2023-08-17 12:02:39,781] INFO [SocketServer listenerType=BROKER, nodeId=1] Enabling request processing. (kafka.network.SocketServer)
[2023-08-17 12:02:39,782] INFO Awaiting socket connections on 0.0.0.0:9092. (kafka.network.DataPlaneAcceptor)
[2023-08-17 12:02:39,783] INFO [BrokerServer id=1] Waiting for all of the authorizer futures to be completed (kafka.server.BrokerServer)
[2023-08-17 12:02:39,784] INFO [BrokerServer id=1] Finished waiting for all of the authorizer futures to be completed (kafka.server.BrokerServer)
[2023-08-17 12:02:39,784] INFO [BrokerServer id=1] Waiting for all of the SocketServer Acceptors to be started (kafka.server.BrokerServer)
[2023-08-17 12:02:39,784] INFO [BrokerServer id=1] Finished waiting for all of the SocketServer Acceptors to be started (kafka.server.BrokerServer)
[2023-08-17 12:02:39,784] INFO [BrokerServer id=1] Transition from STARTING to STARTED (kafka.server.BrokerServer)
[2023-08-17 12:02:39,785] INFO Kafka version: 3.5.0 (org.apache.kafka.common.utils.AppInfoParser)
[2023-08-17 12:02:39,785] INFO Kafka commitId: c97b88d5db4de28d (org.apache.kafka.common.utils.AppInfoParser)
[2023-08-17 12:02:39,785] INFO Kafka startTimeMs: 1692298959784 (org.apache.kafka.common.utils.AppInfoParser)
[2023-08-17 12:02:39,786] INFO [KafkaRaftServer nodeId=1] Kafka Server started (kafka.server.KafkaRaftServer)
```

Kafka Server running on Ubuntu and WSL 2

Produce and consume some messages

In this step, you open two terminal windows, one to run a producer and another to run a consumer.

Open another terminal session and run the `kafka-topics` command to create a Kafka topic named `demo-messages`:

Copy

```
kafka_2.13-3.5.0
```

Run the `kafka-console-producer` command to start producing events to the topic:

Copy

```
bin/kafka-console-producer.sh --topic demo-messages --bootstrap-server localhost:
```

When you’re prompted, type a few lines of text to produce some events:

Copy

```
>first message
>second message
>third message
```

Open another terminal session and run the following command to start consuming events:

Copy

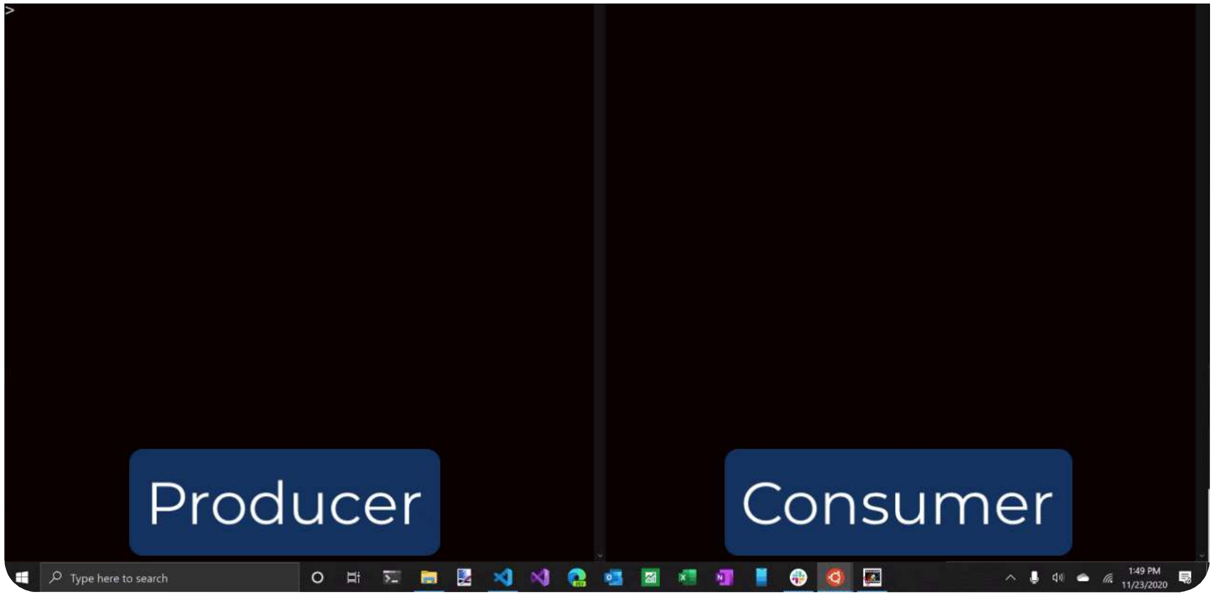
```
cd kafka_2.13-3.5.0
```

Your output should resemble this:

Copy

```
first message
second message
third message
```

Arrange the producer and consumer terminal windows to be side by side. In the producer terminal, type a few more messages, and watch as they appear in the consumer terminal.



Stop Kafka

When you’re done experimenting with Kafka, follow these steps to exit the Kafka environment:

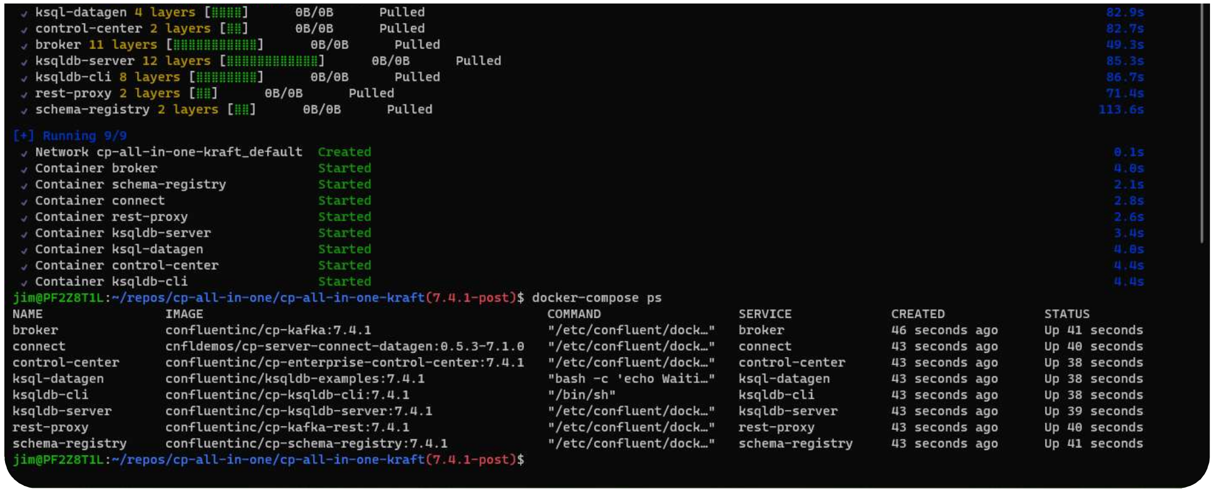
1. Stop the consumer and producer clients with Ctrl+C
2. Stop the Kafka Server with Ctrl+C
3. Run the following command to clean up:

```
rm -rf /tmp/kafka-logs /tmp/zookeeper /tmp/kraft-combined-log
```

Kafka and Windows: The good, the bad, and the ugly

There are lots of Kafka-on-Windows tutorials, but most make the mistake of running Kafka directly on the JVM on Windows. Superficially, this appears to work, but there are limitations: Kafka uses specific features of POSIX to achieve high performance, so emulations—which happen on WSL 1—are insufficient. For example, the broker will crash when it rolls a segment file. Always run Kafka on Windows in a Linux environment backed by WSL 2.

Another approach that works well is to run Kafka in Docker containers. Docker Desktop for Windows has been updated to use the WSL 2 back end, so Docker works exactly as it does on native Linux, without needing to spin up an entire VM. If you want to give this approach a go, try it out using the Confluent Platform demo.



```
 ✓ ksql-datagen 4 layers [####]      0B/0B      Pulled                                                                                         82.9s
 ✓ control-center 2 layers [##]      0B/0B      Pulled                                                                                         82.7s
 ✓ broker 11 layers [###########]      0B/0B      Pulled                                                                                       49.3s
 ✓ ksqldb-server 12 layers [############]      0B/0B      Pulled                                                                               85.3s
 ✓ ksqldb-cli 8 layers [########]      0B/0B      Pulled                                                                                       86.7s
 ✓ rest-proxy 2 layers [##]      0B/0B      Pulled                                                                                             71.4s
 ✓ schema-registry 2 layers [##]      0B/0B      Pulled                                                                                       113.6s

[+] Running 9/9
 ✓ Network cp-all-in-one-kraft_default  Created                                                                                                 0.1s
 ✓ Container broker                     Started                                                                                                 4.0s
 ✓ Container schema-registry            Started                                                                                                 2.1s
 ✓ Container connect                    Started                                                                                                 2.8s
 ✓ Container rest-proxy                 Started                                                                                                 2.6s
 ✓ Container ksqldb-server              Started                                                                                                 3.4s
 ✓ Container ksql-datagen               Started                                                                                                 4.0s
 ✓ Container control-center             Started                                                                                                 4.4s
 ✓ Container ksqldb-cli                 Started                                                                                                 4.4s
jim@PF2Z8T1L:~/repos/cp-all-in-one/cp-all-in-one-kraft(7.4.1-post)$ docker-compose ps
NAME                IMAGE                                             COMMAND                  SERVICE             CREATED             STATUS
broker              confluentinc/cp-kafka:7.4.1                       "/etc/confluent/dock…"   broker              46 seconds ago      Up 41 seconds
connect             cnfldemos/cp-server-connect-datagen:0.5.3-7.1.0   "/etc/confluent/dock…"   connect             43 seconds ago      Up 40 seconds
control-center      confluentinc/cp-enterprise-control-center:7.4.1   "/etc/confluent/dock…"   control-center      43 seconds ago      Up 38 seconds
ksql-datagen        confluentinc/ksqldb-examples:7.4.1                "bash -c 'echo Waiti…"   ksql-datagen        43 seconds ago      Up 38 seconds
ksqldb-cli          confluentinc/cp-ksqldb-cli:7.4.1                  "/bin/sh"                ksqldb-cli          43 seconds ago      Up 38 seconds
ksqldb-server       confluentinc/cp-ksqldb-server:7.4.1               "/etc/confluent/dock…"   ksqldb-server       43 seconds ago      Up 39 seconds
rest-proxy          confluentinc/cp-kafka-rest:7.4.1                  "/etc/confluent/dock…"   rest-proxy          43 seconds ago      Up 40 seconds
schema-registry     confluentinc/cp-schema-registry:7.4.1             "/etc/confluent/dock…"   schema-registry     43 seconds ago      Up 41 seconds
jim@PF2Z8T1L:~/repos/cp-all-in-one/cp-all-in-one-kraft(7.4.1-post)$
```

Apache Kafka and Confluent Platform running in Docker containers on Ubuntu and WSL 2

You're just getting started!

Although Kafka provides an event streaming platform to build your applications on, you'll want to take advantage of the broader ecosystem of components—like ksqlDB, Confluent Schema Registry, and Confluent Control Center—all provided as part of Confluent Platform. At the moment, Confluent Platform is supported for *experimentation only* on Windows, not for production or development environments.

Now that you have Kafka installed, you'll want to learn more about it, try out the numerous tutorials, and join the community! Don't forget that Apache Kafka has many APIs—including the producer and consumer but also Kafka Streams and Connect.

Kafka on Windows? What made this possible?

You may recall a time when Linux was anathema to Microsoft. Back in 2001, Microsoft CEO Steve Ballmer famously called Linux a “malignant cancer,” but he has since come around to “loving” it. Microsoft’s current CEO Satya Nadella seems intent on making it a first-class citizen in the Microsoft ecosystem, which means that a new era has arrived for software developers on the Windows platform.

When the Windows Subsystem for Linux (WSL 1) was released in 2016, it became possible to run a real Linux dev environment in a Linux shell, while retaining the familiar Windows UX around the shell. Even File Explorer was integrated nicely with the Linux file system.

The big drawbacks are that WSL 1 emulates a Linux kernel, and it runs in a full VM. The first means processes that require a native kernel, like Docker, can’t run. The second means that WSL 1 consumes a lot of resources. WSL 1 was not sufficient to run Kafka reliably.

But Microsoft delivered WSL 2 in 2019, and it’s a whole new world. They fixed the two biggest limitations, so WSL 2 runs a real Linux kernel, and the kernel on a subset of Hyper-V features, not in a full VM. For details, see