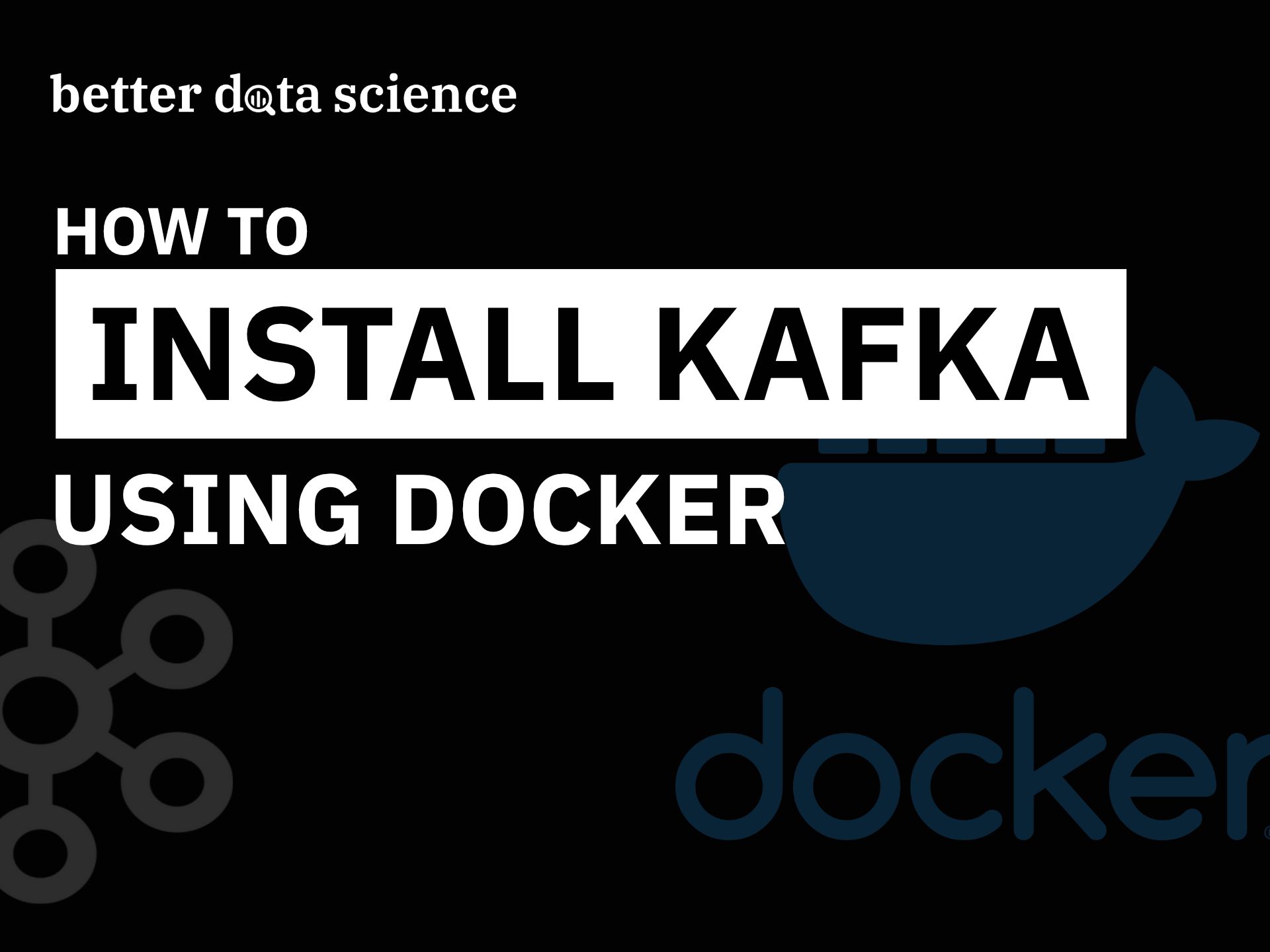
**How to Install Apache Kafka Using Docker — The Easy Way**



**And how to create your first Kafka Topic. Video guide available.**

In a world of big data, a reliable streaming platform is a must. That’s where Kafka comes in. And today, you’ll learn how to install it on your machine and create your first Kafka topic.

Want to sit back and watch? I’ve got you covered:

Today’s article covers the following topics:

* Approaches to installing Kafka
* Terminology rundown — Everything you need to know
* Install Kafka using Docker
* Connect to Kafka shell
* Create your first Kafka topic
* Connect Visual Studio Code to Kafka container
* Summary & Next steps

**Approaches to installing Kafka**

You can install Kafka on any OS, like Windows, Mac, or Linux, but the installation process is somewhat different for every OS. So, instead of covering all of them, my goal was to install Kafka in a virtual machine and use Linux Ubuntu as a distribution of choice.

But, since I’m using a MacBook with the M1 chip, managing virtual machines isn’t all that easy. Attaching ISO images to VirtualBox just fails. If anyone knows the solution, please let me know in the comment section below.

So instead, you’ll use Docker. I think it’s an even better option, since you don’t have to install the tools manually. Instead, you’ll write one simple Docker compose file, which will take care of everything. And the best part is — it will work on any OS. So, if you are following this article on Windows or Linux, everything will still work. You only need to have Docker and Docker compose installed.

Refer to the video if you need instructions on installing Docker.

**Terminology rundown — Everything you need to know**

This article is by no means an extensive guide to Docker or Kafka. Heck, it shouldn’t even be your first article on these topics. The rest of the section will give only high-level definitions and overviews. There’s a lot more going in to these concepts that are way beyond the scope of this article.

**Kafka** — Basically an event streaming platform. It enables users to collect, store, and process data to build real-time event-driven applications. It’s written in Java and Scala, but you don’t have to know these to work with Kafka. There’s also a Python API.

**Kafka broker** — A single Kafka Cluster is made of Brokers. They handle producers and consumers and keeps data replicated in the cluster.

**Kafka topic** — A category to which records are published. Imagine you had a large news site — each news category could be a single Kafka topic.

**Kafka producer** — An application (a piece of code) you write to get data to Kafka.

**Kafka consumer** — A program you write to get data out of Kafka. Sometimes a consumer is also a producer, as it puts data elsewhere in Kafka.

**Zookeeper** — Used to manage a Kafka cluster, track node status, and maintain a list of topics and messages. Kafka version 2.8.0 introduced early access to a Kafka version without Zookeeper, but it’s not ready yet for production environments.

**Docker** — An open-source platform for building, deploying, and managing containers. It allows you to package your applications into containers, which simplifies application distribution. That way, you know if the application works on your machine, it will work on any machine you deploy it to.

You now have some basic high-level understanding of the concepts in Kafka, Zookeeper, and Docker. The next step is to install Zookeeper and Kafka using Docker.

**Install Kafka using Docker**

You will need two Docker images to get Kafka running:

* [wurstmeister/zookeeper](https://hub.docker.com/r/wurstmeister/zookeeper/)
* [wurstmeister/kafka](https://hub.docker.com/r/wurstmeister/kafka/)

You don’t have to download them manually, as a docker-compose.yml will do that for you. Here’s the code, so you can copy it to your machine:

version: '3'

services:

zookeeper:

image: wurstmeister/zookeeper

container\_name: zookeeper

ports:

- "2181:2181"

kafka:

image: wurstmeister/kafka

container\_name: kafka

ports:

- "9092:9092"

environment:

KAFKA\_ADVERTISED\_HOST\_NAME: localhost

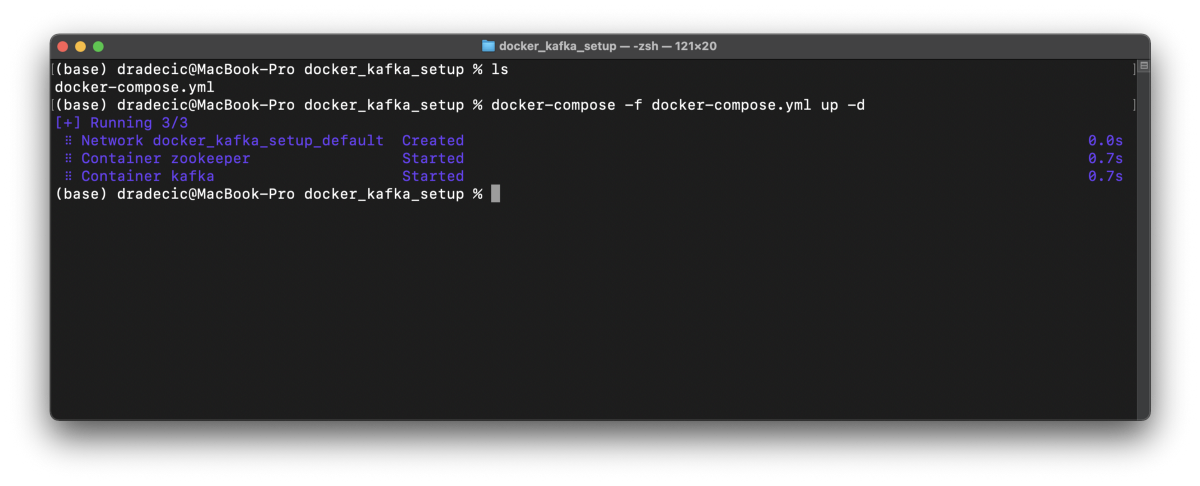
KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

Make sure to edit the ports if either 2181 or 9092 aren’t available on your machine. You can now open up a Terminal and navigate to a folder where you saved docker-compose.yml file. Execute the following command to pull the images and create containers:

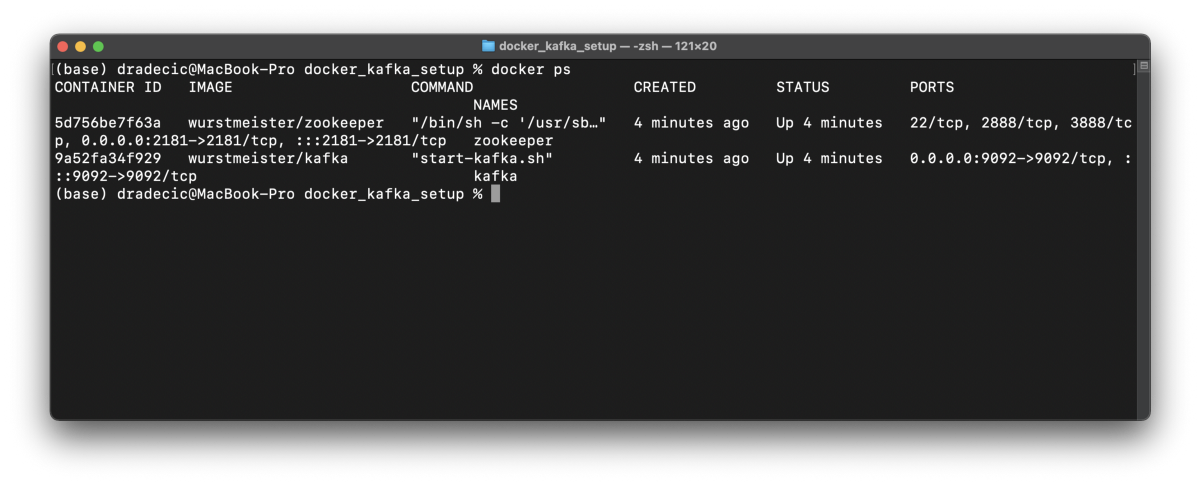
docker-compose -f docker-compose.yml up -d

The -d means both Zookeeper and Kafka will run in the background, so you’ll have access to the Terminal after they start.

You should now see the download and configuration process printed to the Terminal. Here’s how mine looks like, but keep in mind — I already have these two configured:

Image 1 — Docker compose for Zookeeper and Kafka (image by author)

And that’s it! You can use the docker ps command to verify both are running:

Image 2 — Docker PS command (image by author)

*But what can you now do with these two containers?* Let’s cover that next, by opening up a Kafka terminal and creating your first Kafka topic.

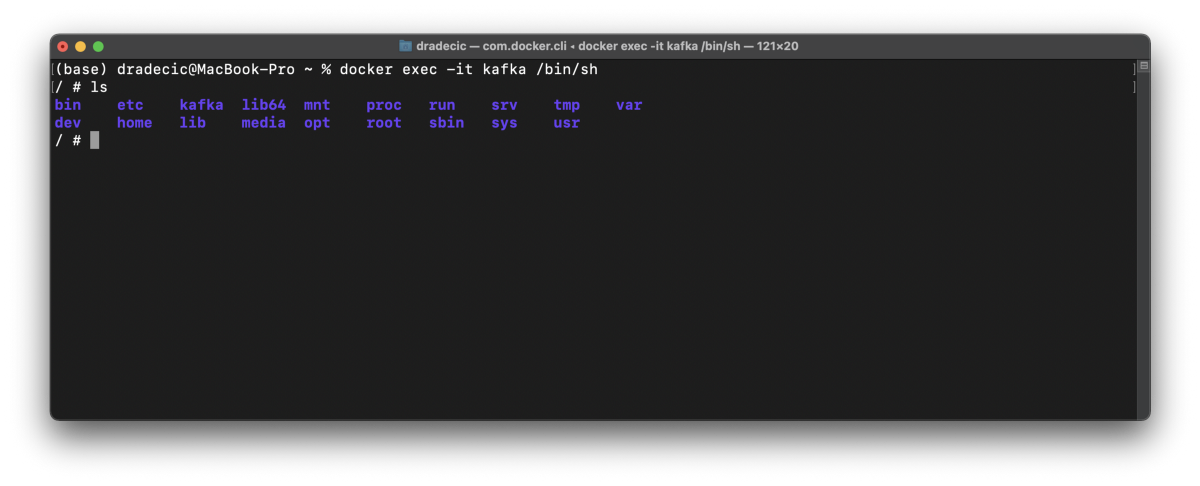
**Connect to Kafka shell**

Once Zookeeper and Kafka containers are running, you can execute the following Terminal command to start a Kafka shell:

docker exec -it kafka /bin/sh

Just replace kafka with the value of container\_name, if you’ve decided to name it differently in the docker-compose.yml file.

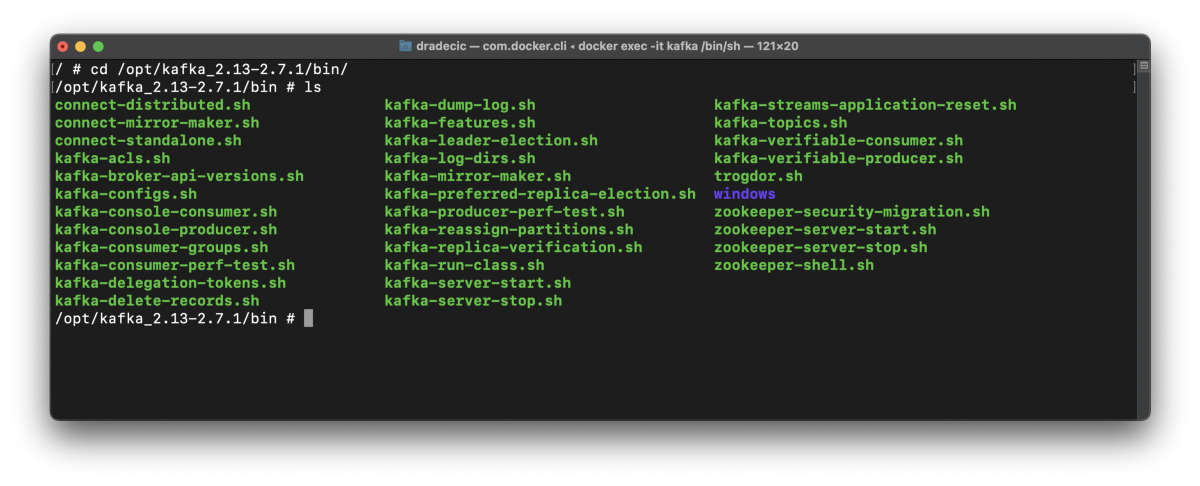
Here’s what you should see:

Image 3 — Connecting to Kafka shell (image by author)

Now you have everything needed to create your first Kafka topic!

**Create your first Kafka topic**

All Kafka shell scripts are located in /opt/kafka\_<version>/bin:

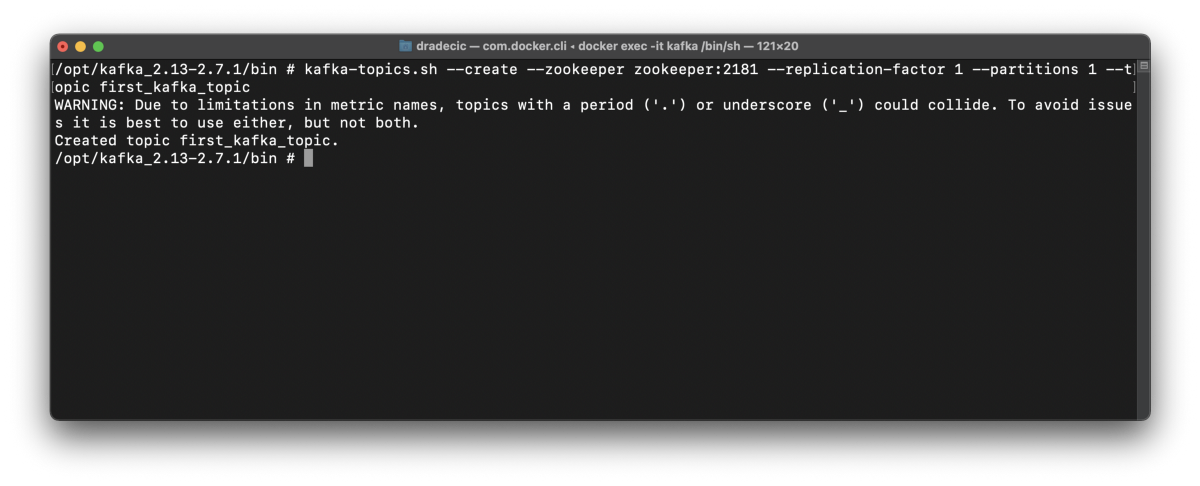
Image 4 — All Kafka shell scripts (image by author)

Here’s the command you’ll have to issue to create a Kafka topic:

kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 --partitions 1 --topic first\_kafka\_topic

Where first\_kafka\_topic is the name of your topic. Since this is a dummy environment, you can keep replication-factor and partitions at 1.

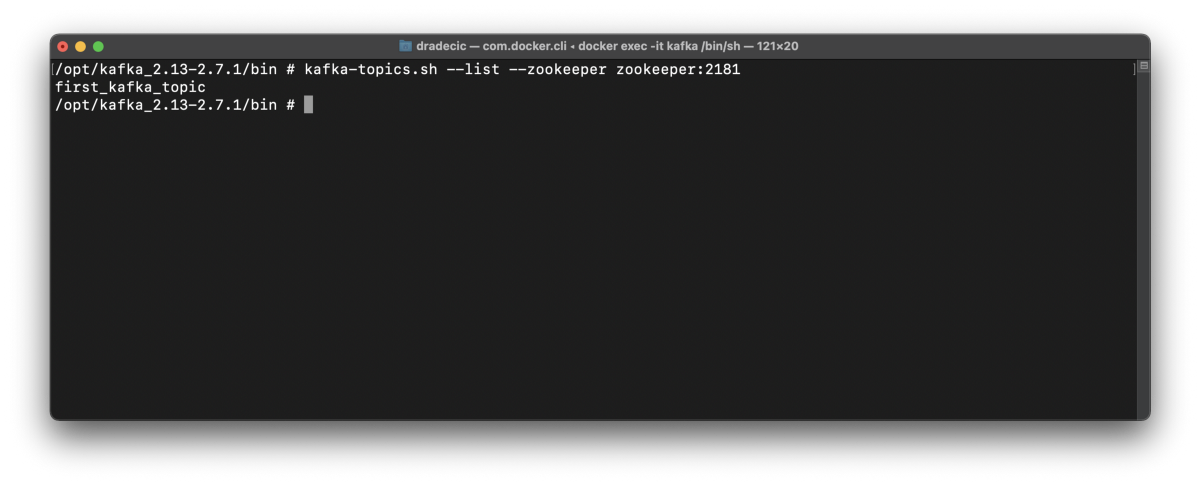
Here’s the output you should see:

Image 5 — Creating a Kafka topic (image by author)

And that’s it! The topic will be created after a second or so. You can list all Kafka topics with the following command:

kafka-topics.sh --list --zookeeper zookeeper:2181

Here’s what it prints on my machine:

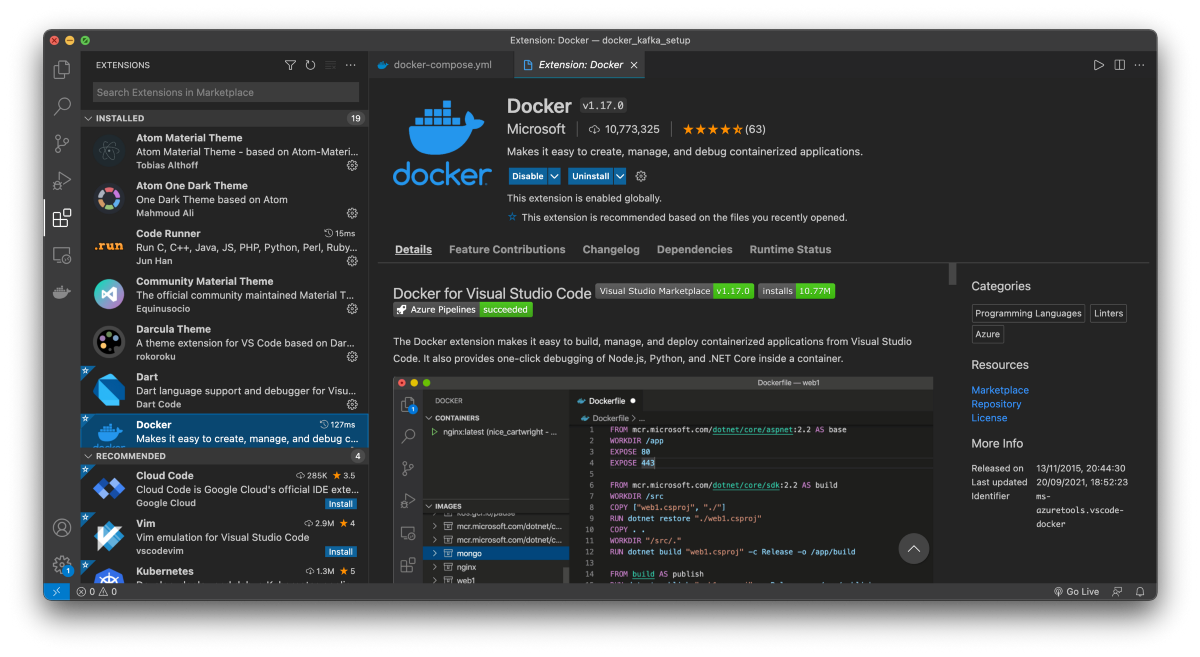
Image 6 — Listing Kafka topics (image by author)

And that’s how you create a Kafka topic. You won’t do anything with it today. The following article will cover how to write Producers and Consumers in Python, but there’s still something else to cover today.

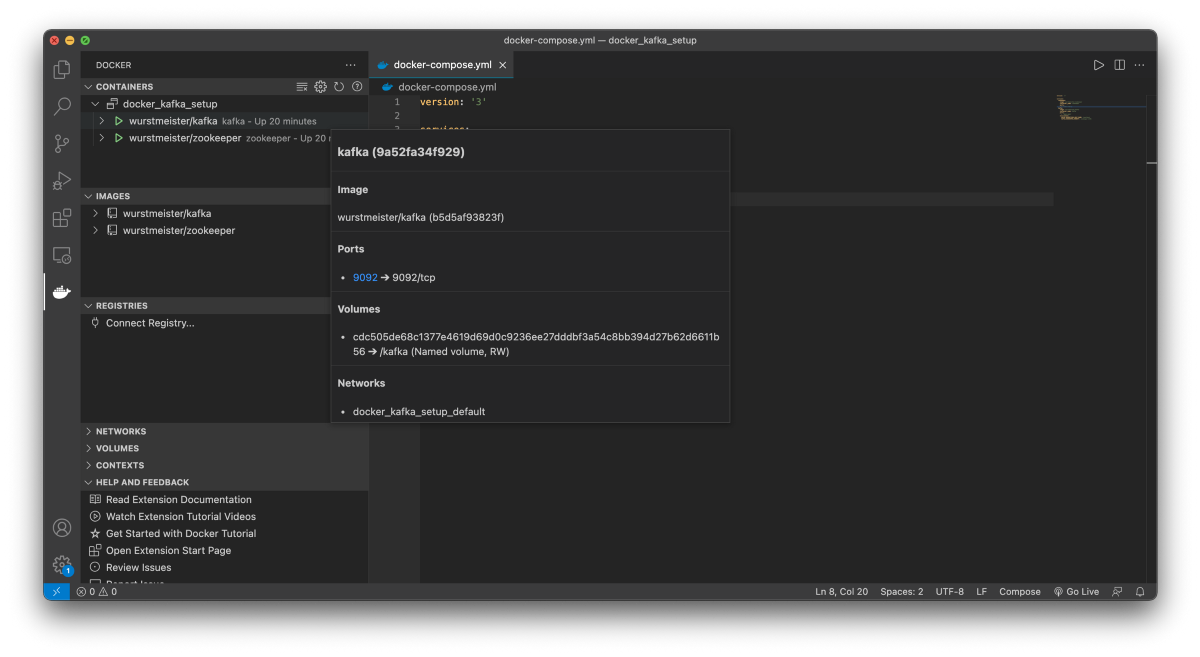
**Connect Visual Studio Code to Kafka container**

No, you won’t write producers and consumers today, but you will in the following article. Manually transferring Python files from your machine to a Docker container is tedious. You can write code directly on the Kafka container with Visual Studio Code.

You will need an official Docker extension by Microsoft, so install it if you don’t have it already:

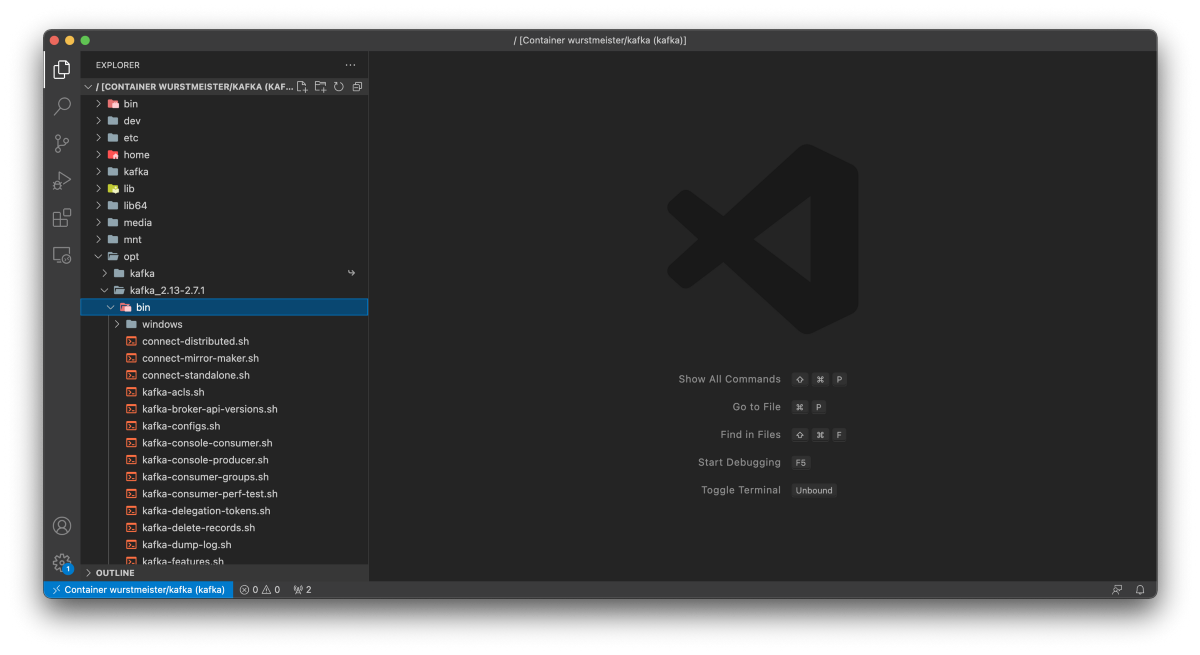
Image 7 — Installing the VSCode Docker extension (image by author)

Once it installs, click on the Docker icon in the left sidebar. You’ll see all running containers listed on the top:

Image 8 — Listing Docker containers (image by author)

You can attach Visual Studio Code to this container by right-clicking on it and choosing the *Attach Visual Studio Code* option. It will open a new window and ask you which folder to open.

Go to *root* (/), and you should see all the folders you saw minutes ago when you were in the shell:

Image 9 — Exploring Kafka container through VSCode (image by author)

*Easy, right?* Well, it is, and it’ll save you much time when writing consumers and producer code.

**Summary & Next steps**

And that does it for today. You went from installing Docker, Kafka, and Zookeeper to creating your first Kafka topic and connecting to Docker containers through Shell and Visual Studio Code.

Refer to the video if you need detailed instructions on installing Docker and attaching Kafka shell through Visual Studio Code.

The following article will cover consumers and producers in Python, so stay tuned if you want to learn more about Kafka.

**Stay connected**

* Follow me on [Medium](https://medium.com/@radecicdario) for more stories like this
* Sign up for my [newsletter](https://mailchi.mp/46a3d2989d9b/bdssubscribe)