DOCK1

Can you Contain yourself??

Elevate your Mainframe Experience with Docker

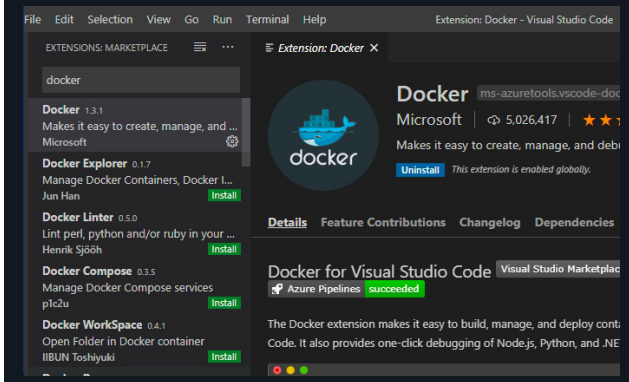
9 steps **1 hour**

THE CHALLENGE

By now, you know how the deep-down mechanics of z/OS work. You know how to allocate a data set from Zowe, from your terminal, through VS Code, through ZOAU... you're at the point where you just want things to happen as quickly and smoothly as possible, and we're about to play around in Docker, an open-source platform to create, deploy and manage virtualized application containers.

BEFORE YOU BEGIN

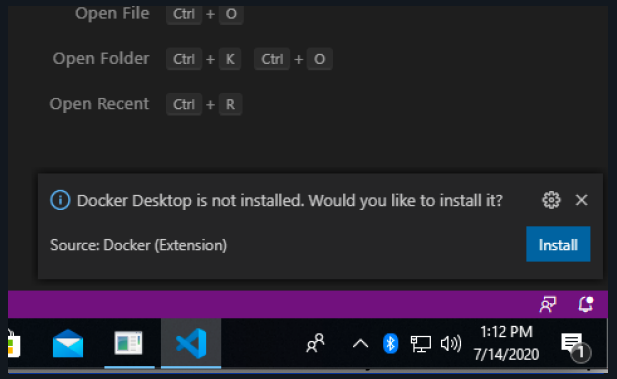
You'll need some essential tools in place – Docker and a couple of VS Code extensions. This will require some setup of software on your workstation and in VS Code – all worth it!



1. INSTALL DOCKER IN VS CODE

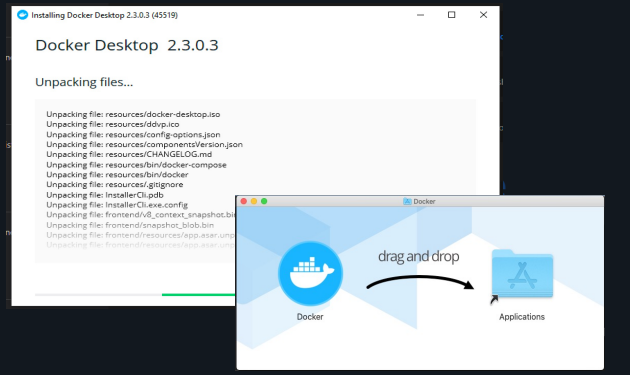
Head into Extensions and look for Docker. There are lots of docker-related extensions, but we're going to start out with just plain old "Docker". This will give VS Code the ability to work with Docker images and containers. The install should be pretty quick, but you're not done yet.

Go into the new Docker icon that was created on the left side and you'll notice that all of the boxes have little yellow "warning" triangles in them, because Docker Desktop isn't installed yet. Let's fix that in the next step.



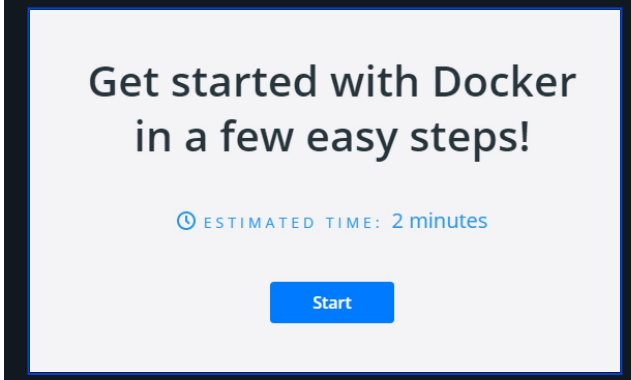
2. INSTALL DOCKER DESKTOP

If this message pops up, click Install. If you missed it, or still have a warning that Docker Desktop isn't installed/running, then head over to Docker Desktop and start the download. This is the more complicated install, and might take up to 30 minutes depending on your computer speed and internet connection.



3. FOLLOW THE PROMPTS

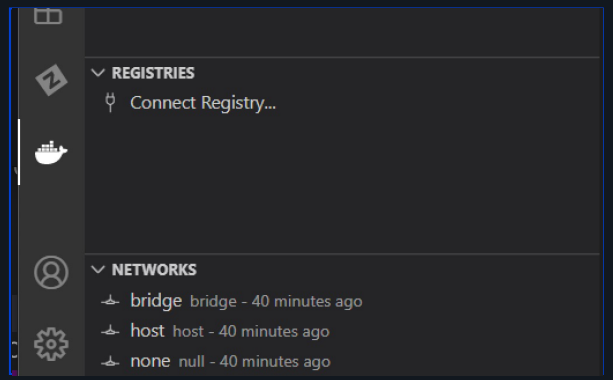
You're going to have to use your best judgement here for a while. Installation should be fairly straightforward. On Windows, you'll run an installer and reboot. On Mac, you'll drag Docker into Applications. A reboot may be required. After the install finishes, launch Docker Desktop and follow the prompts.



4. TAKE THE TOUR (OPTIONAL)

Want to play around with a sample container and try out a few commands before we get started? There's a built-in demo that might help you get some of the Docker concepts.

Already done, or feeling like you're pretty solid on Docker? Then move along to the next step, but make sure to keep Docker Desktop running. We'll verify this in Step 5.

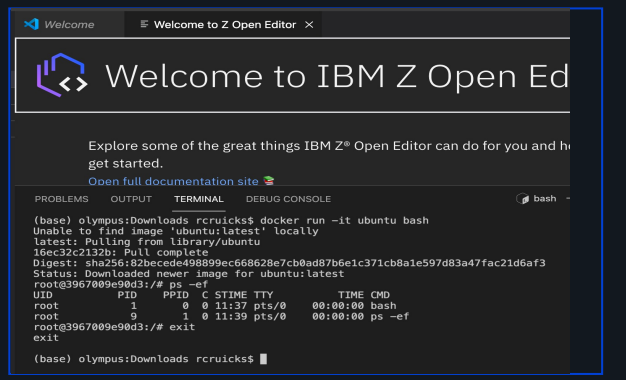


5. LOAD UP AN IMAGE

When you relaunch VS Code, click on the Docker icon on the left side (it's that iconic whale with the containers on its back). You should see no warning triangles, but instead something similar to the screenshot above.

Go into your Terminal and enter this command to run a docker image as a container, and give us an interactive shell within it.

docker run -it ubuntu bash



6. RUNNING, JUST LIKE THAT

Since we don't already have this image, it will need to download all of the pieces. When done, you'll notice that we now have:
1) A container
2) An image
3) The terminal prompt is a window into the docker container

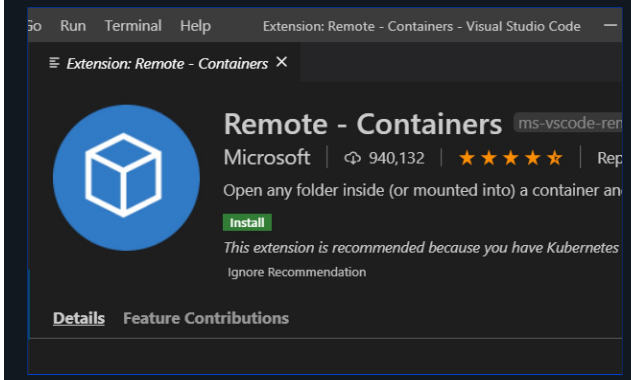
This is all... a little mind-bending at first. It's acting like a virtual machine running on your computer, but it's really just the parts you will need to run Ansible.

We'll help make this a little more clear by installing one more extension, catch that in the next step.

I CAN'T GET DOCKER RUNNING. DO I NEED IT?

Docker is a technology which allows solutions that run in a standardized environment (basically, anything that runs Docker) to be picked up and made to run anywhere else. This means that every little file, library, piece of code, even full software packages and operating systems can be packaged up into a Docker Container and made available for distribution. We're making this challenge a little bit easier by making the Ansible environment pre-packaged in a Docker Container.

If you can't get it installed, don't want to, or want to keep your options open, you can install and use Docker on a Linux image you created in the Linux-based challenges. Or, bypass Docker altogether and install Ansible right on most operating systems. For this guide, however, we'll follow the Docker Desktop / VS Code method, as it should work for most people.

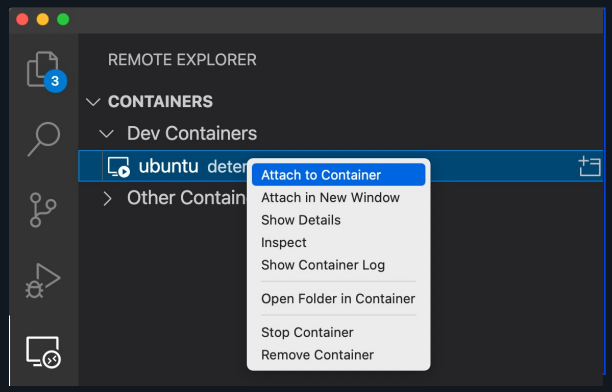


7. INSTALL REMOTE - CONTAINERS

Head into the Extensions screen (the 4-box icon) to search and install Remote - Containers. This will let us launch an entirely separate VS Code window for each container we're working on.

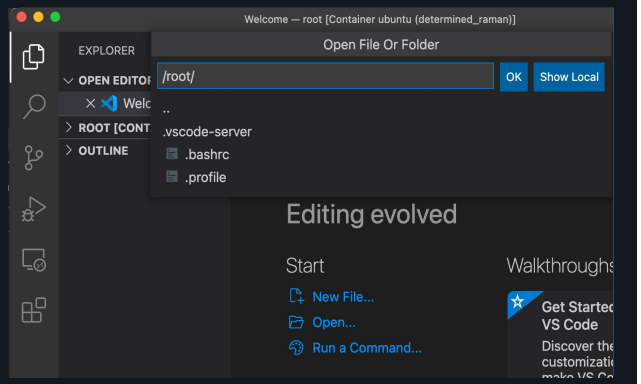
You can skip this if you have a method you prefer, but we'll be showing steps using the extension, to better illustrate the different environments.

Once done, give VS Code a restart (just to catch everything up) and move to Step 8.



8. LAUNCH REMOTE EXPLORER

Once back in, find the new icon on the left side for Remote Explorer, and find your Ubuntu container in it which we just pulled down. (You may also need to redo the command from Step 5 if you restarted VS Code)
Right-click on it and select "**Attach to Container**". This will launch a new VS Code window, which gives you a view into what's happening within this container image. This is the container view, not your computer's regular filesystem. Notice the green box at the bottom of the window showing you what the view is, and also notice that the Zowe icon is absent.



9. OPEN ROOT

Click on the Explorer button, then click "Open ...", and enter "/root/" as the folder you want to open. You should wind up with a listing of files like the one in the screenshot above.

Now you're ready to tackle all manner of containers based activities – you can use this setup to get hands-on experience with Ansible.
You can mark your challenge complete by submitting the **CHKADOCK** job in ZXP.PUBLIC.JCL

WHY ALL THE PREP WORK?

Getting a program to work once is good. Getting it to work hundreds, if not, thousands of times after that, the exact same way, takes careful orchestration. In the past, that meant that a person, or a team of people had to go and configure a whole bunch of systems, and carefully set them up so that a program could run on them. They would manage every step of the process, and doubling the amount of programs running meant everyone had to work twice as hard. That's the old way.

Docker containers, and orchestration that happens in solutions like Ansible and Kubernetes means that scaling up in capability doesn't mean adding complexity at every step. It's an important revolution in Enterprise IT that not only produces more capable solutions, but frees up staff to work on other tasks, like coming up with the next greatest thing, instead of setting up yet another system.

NEXT UP...

We've set it all up, we might as well use it. The Ansible modules depend on the use of docker containers to build and execute "playbooks" – you should be all set to tackle these now.