Running Minecraft In Production - Part 1

The_Sudo

HOW TO RUN MINECRAFT ON DOCKER | RUNNING MINECRAFT IN PRODUCTION PART 1

https://www.youtube.com/watch?v=NUlUk6hc-mQ&list=PLZlXrZGK4hBiJx_NasfmFEuLVB4oLKIul

Install WSL

terminal: - -wls install

check version: wsl -l -v

GitHub:

- Tag: a versión of the container image
- Pull: download the conainer image down to local PC but not run it
- Run: download the container down to local PC and then run it

Docker Desktop:

Ports in Docker and containers are a mapping of host system port to container port.

Host port:

First number is system port, the port on the computer that will map to the port of the container.

Second number is port of the container. We can't change ":25565/tcp" but we can choose which port on our computer that this Docker container is mapped to.

I can assign any port. This is the port on my network that I would forward the traffic through in my router for others to connect to this server.

Cannot use the same port for multiple applications

Docker Desktop main page shows all of the running containers and their statuses, their health

Installation Process:

1. Search for Minecraft image
2. Choose Run
3. Open Optional Settings: Run a new container setup
4. Skip volumes and variables for now.
5. After adding name and port hit "Run"

Docker Container stopped (“exited”)

Need to accept the Minecraft EULA

Container names need to be unique. Assigning a name is not a requirement but it will automatically receive a random name

Using Docker CLI: in terminal

- **docker ps** : shows containers running
- **docker ps** -a : shows all containers including exited ones
- **docker logs** insertcontainername : view logs for container and why container exited
- **docker logs** insertcontainername **-f** : will show the continuos running output, watch logs in real time
    a. **control+c** to exit / cancel
- **control+l** : clear screen
- **docker run** : run container
- **docker compose** : check if Docker Compose is installed and get list of commands
- **docker stop [CONTAINER_NAME_OR_ID]** : stop specific container
- **docker stop --timeout** : stop container in timeout countdown (not specified will do default, usually 10 seconds)
- docker stop --timeout 30 my-web-app : (example specific timeout time and container name)

Full Docker CLI minecraft server command:

docker run -d --name insertcontainername **-p 25565:25565 “-v F:**mcdata**:/data -e EULA=true itzg/minecraft-server:latest**

Pass a flag: dash + letter

a. **-d** : Need to pass the flag “ -d ” which stands for <u>detached</u> which means: Please run Docker container in the background and detach it from the current terminal session.
b. **- -name** : chosen container name
c. **-p** : port mapping
d. **– e** : environment variables. Can pass as many as wanted
e. Pass actual namespace and container image name as it exists on Docker Hub
f. Add tag (latest). If tag not provided it will be latest by default
g. “**- v**” : map or mount a volume on your pc to a volume within a container. F: = folder location

-

“ **-d** “ : If not detatched then Docker will run in the foreground and the terminal will be taken over by the logs from the minecraft container.

Need to insert comando to accept user and license agreement for Minecraft:

ENVIRONMENT VARIABLES.
Find necessary variables in DockerHub container image page, where the Documentation should be.

Whoever creates the container image is the one that decides what variables get to pass into that container.

Variables in command are specified by "-e "

" docker run -d -it -p 25565:25565 -e EULA=TRUE itzg/minecraft-server "

Installation Process:

1. Search for Minecraft image
2. Choose Run
3. Open Optional settings
4. "Run a new container" setup
    a. Define Container name
    b. Define Port
    c. Environment Variables:
        - EULA = True
        - ONLINE_MODE = False
5. Run

Inspect Docker Container

Docker Desktop> Containers > insertcontainername > Inspect tab

Docker Desktop> Containers > insertcontainername > Terminal tab

rcon-cli : Remote console (RCON) is a TCP/IP-based protocol that allows server administrators to remotely execute commands

Whitelist command: whitelist add playername

RCON RUNNING in logs : now can connect to Minecraft server

Now open Minecraft launcher

The version in launcher needs to be the same as the version launched in the Minecraft server container

Check logs:

[Server thread/INFO]: Starting minecraft server version 1.21.10

In Minecraft launcher:

Multiplayer > Add Server > Server Address: localhost:25565 > Done

Server Address will be the port mapped or bound the container to the PC

localhost:25565

Confirm the correct IP address

- Local IP: On the server computer, open the command prompt and type ipconfig (Windows) or ip a (Linux) to find the local IPv4 address. This is the address you should use to connect from another computer on the same network.
- localhost: If connecting from the *same* computer that the server is running on, use localhost or 127.0.0.1 as the server address.

Volumes

Docker Desktop > Volumes > insertcontainername > Container in-use:

Target means that there is a directory (named "/data")

Container terminal: CD to the root and do an LS I can see that /data is a directory that contains all the data from the minecraft server.

If this container disappears, al lof the data within that container also disappears.

We make our data persistent through volumes.

Docker Container Documentation Page > Data Directory

Section explains how all the data within the container is set up.

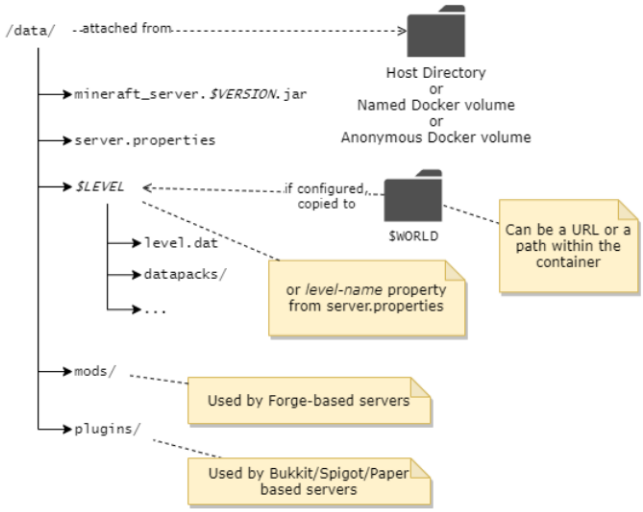
It says we need to attatch /data/ to our host directory somwhere on our computer

https://hub.docker.com/r/itzg/minecraft-server

https://docker-minecraft-server.readthedocs.io/en/latest/

Data directory

Everything the container manages is located under the **container's** `/data` path, as shown here:



Command flag "**- v**" : map or mount a volume on your pc to a volume within a container

Attaching data directory to host filesystem

In most cases the easiest way to persist and work with the minecraft data files is to use bind mounts with the `-v` argument to map a directory on your host machine to the container's `/data` directory. In the following example, the path `/home/user/minecraft-data` **must be** a directory on your host machine:

```
-v /home/user/minecraft-data:/data
   ------------------------ -----
    |                       |
    |                       +-- must always be /data
    |
    +-- replace with a directory on your host machine
```

Steps:

1. Create folder where desired on computer in File Explorer
2. Copy folder path / location
3. Run new container

4. > Optional settings > Volumes > Host path: folder location
5. > Optional settings > Volumes > **Container path: "/data"** (defined in Documentation)
6. Fill out rest of Optional settings > Run

Now Minecraft files will populate inside designated folder and will be saved even if container is lost.
Server data will be saved on computer hard drive.

Process for Docker CLI:

- **docker run command with additional flag "-v F:**mcdata**:/data**" (F: = folder location example)
  = path on local host and path within the container that we want to map it to

(Recap:)

Installed docker desktop, installed minecraft container image, ran container, configured container with different environment variables and also mapped a file system on local computer to data directory within Minecraft container

Docker Compose :

Fixes problema of Constantly destroying and recreating containers and having to rewrite all Optional settings.

terminal: **" docker compose "** : check if Docker Compose is installed and get list of commands

TheSudo uses Visual Studio Code by choice when using Docker Compose.

You can make a new folder to save container settings **(/mc-docker** example).

Visual Studio Code:

- New file in designated folder
- name: "**docker-compose.yml**"

yml = Yaml = another markup language

Docker Container Documentation Page > Intro > Using Docker Compose section

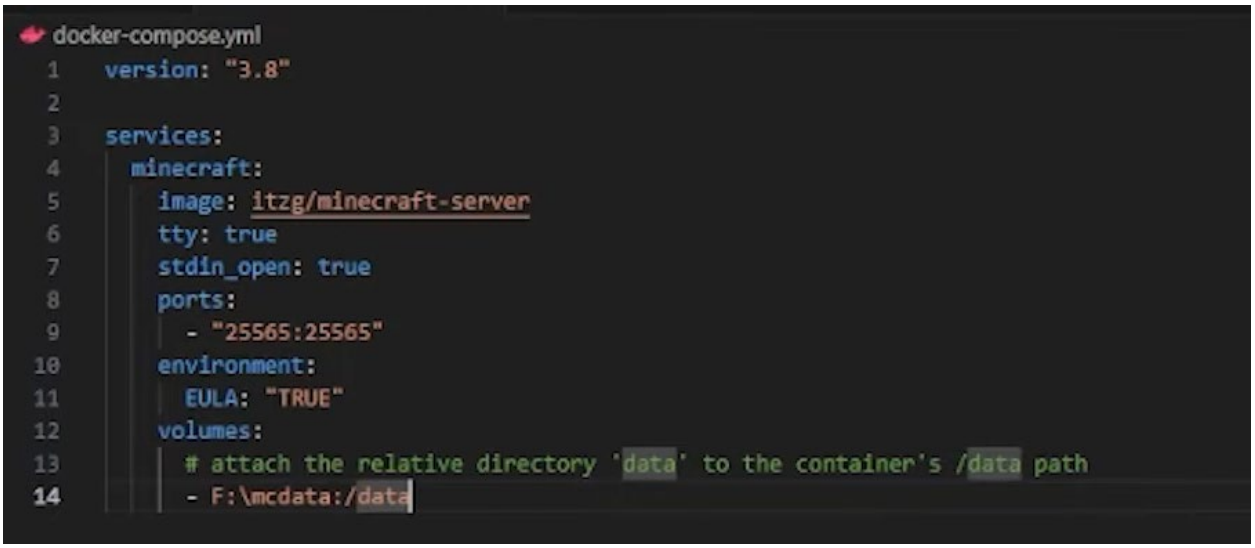
image creator provided basic docker compose file to start. Copy-paste settings into Visual Studio Code

"version" is a Docker compose thing, nothing to do with minecraft

“services” : where we define all of the services that we want running

Typically containers are referred to as services or microservices

Previous Docker Desktop / Docker CLI configuration written into a file:

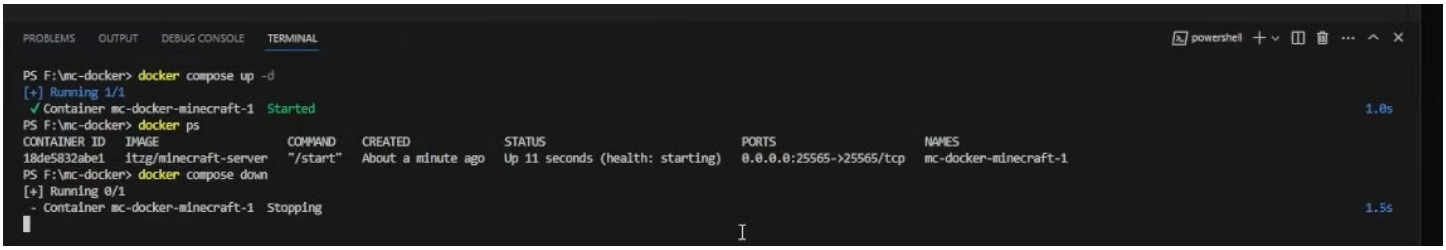


Visual Studio Code:

- Terminal (will open a PowerShell terminal withing VS Code below file)
- Ls : to check if in right directory within terminal
- command: “ **docker compose up -d** “
    - If running from the same place the Docker compose .yml file exists, no need to provide the path to Docker compose file
    - Provide file path if Docker compose yml. file is in different directory than the one running Docker Compose up from.
        - Use LS and CD to move to right directory in terminal. Folders with spaces in name give errors (?)

- command “ **docker compose stop** “ : stops container
- command “ **docker compose down** “ : shuts down and deletes container

https://medium.com/@laurap_85411/docker-compose-stop-vs-down-e4e8d6515a85



Now in Docker Desktop > Containers, the container symbol is different than before because it is now a **Docker Compose Stack**.

Typically with Docker Compose, multiple containers are deployed at once, defining multiple services.

When Docker Compose runs, all the container services show under the same stack.

Docker Compose will also be used to launch Prometheus and Grafana containiers along with Minecraft containers.

- Restarting Docker Compose : **Restart Policy**

Automating response if something goes wrong with container.

command: " **restart: on-faliure:3** "

Specifying restart policy: if container fails, how many attempts should it do to restart before giving up (3 tries). If amount of attempts not specified, it will restart forever.

- **Chaos engineering**: Testing restart policy without EULA

1. Comment out environment section and environment variable with pound symbol (#). Container will fail to start if it doesn't pass the EULA.

   - #environment:
     #EULA: "TRUE"

2. Erase file called “EULA” within the /data directory saved on computer. This file is where whether or not the end user license agreement is accepted is being tracked.

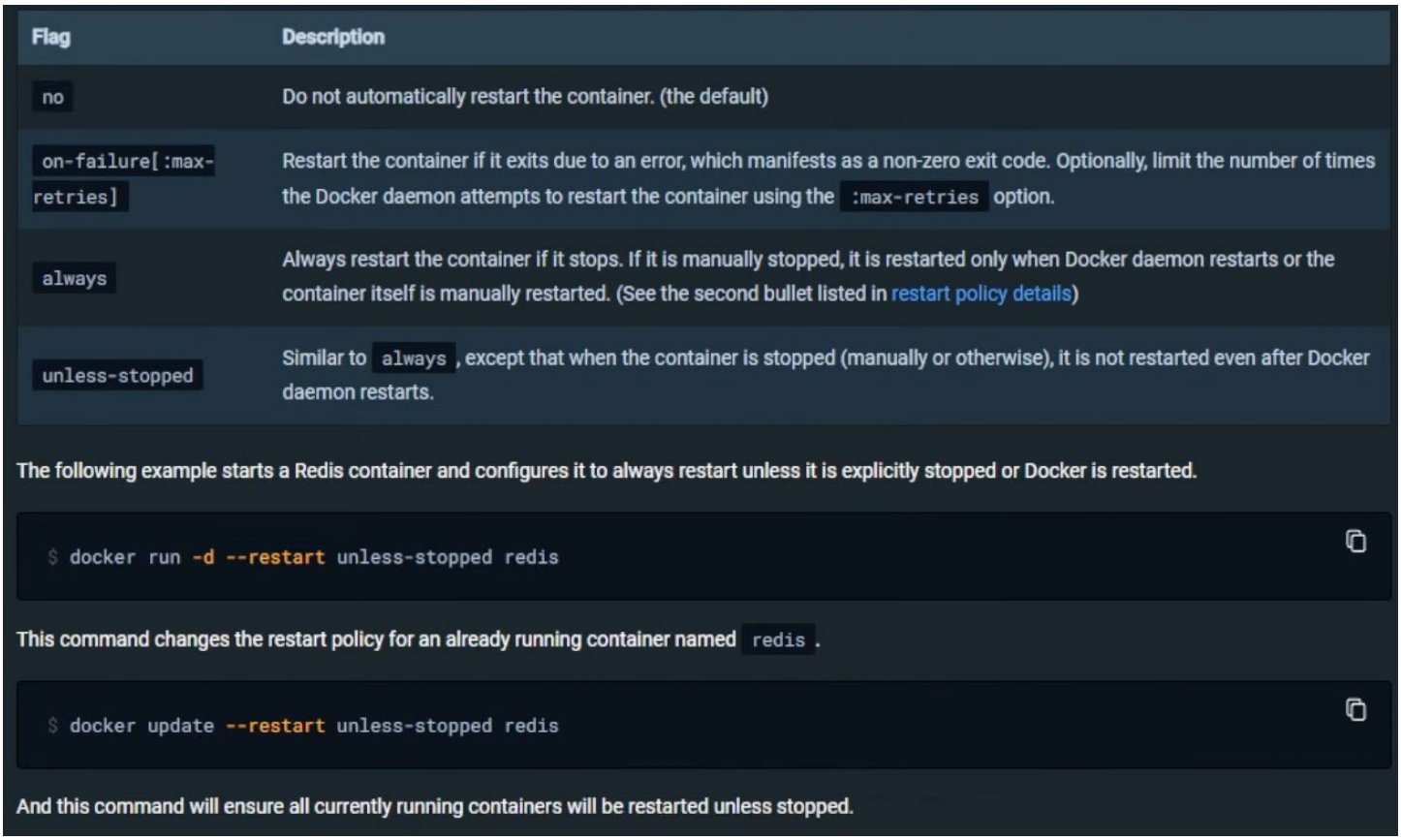
Now container will fail to start and attempt to restart.

Docker Desktop > Containers > check container log to see restart attempts.

Docker CLI command : - -restart on-faliure

docker run -d --name insertcontainername -p 25565:25565 “-v F:\mcdata:/data -e EULA=true --restart on-faliure itzg/minecraft-server:latest

Options for restart policies are also in the image documentation:



Flag	Description
`no`	Do not automatically restart the container. (the default)
`on-failure[:max-retries]`	Restart the container if it exits due to an error, which manifests as a non-zero exit code. Optionally, limit the number of times the Docker daemon attempts to restart the container using the `:max-retries` option.
`always`	Always restart the container if it stops. If it is manually stopped, it is restarted only when Docker daemon restarts or the container itself is manually restarted. (See the second bullet listed in restart policy details)
`unless-stopped`	Similar to `always`, except that when the container is stopped (manually or otherwise), it is not restarted even after Docker daemon restarts.