

Sidebar

Programming · VS Code Docker Cheat Sheet

Working with Docker in VS Code through the Container Tools extension: Dockerfile and Compose editing, the Container Explorer, Compose, tasks, debugging and settings. Checked against Container Tools 2.5.2 installed here. Sheet 22 in the VS Code series. The Remote and Containers sheet covers dev containers, which are a different feature.

WHICH EXTENSION? (AND WHAT IS INSTALLED)

Container Tools `ms-azuretools.vscode-containers` (version 2.5.2 here): the extension that does the work; needs a recent VS Code (1.109 or later)

Docker (`ms-azuretools.vscode-docker`) Now just an extension pack (2.0.0 here) that installs Container Tools; it can be uninstalled without losing anything

Docker DX (`docker.docker`) Docker's own language service for more advanced Dockerfile, Compose and Bake IntelliSense; recommended alongside Container Tools

Docker or Docker Desktop Must be installed and on your PATH; the extension drives the `docker` program

Recognised files `Dockerfile`, `Dockerfile.*`, `*.dockerfile`, `Containerfile` and `.dockerignore` / `.containerignore`

CREATING & EDITING DOCKER FILES

Containers: Add Docker Files to Workspace... Command `.dockerignore` or Java

Containers: Add Compose Files to Workspace... Command

Ctrl+Shift+M Problems panel: shows errors found in Dockerfile

containers.enableComposeLanguageService IntelliSense and `true`

docker.languageserver.diagnostics.* Checks such as `instructionCmd`, `instructionEnt`, `instructionWork`, `warning` (default)

docker.languageserver.formatter.ignoreMultilineInstru

[dockercompose] defaults shipped Two-space indents, space and suggestions inside

CONTAINER EXPLORER (ACTIVITY BAR)

Containers Right-click: Start, Stop, Restart, Remove, View Logs, Attach Shell, Inspect, Stats, Download, Open in Integrated or External Browser, Filter, Prune

Images Right-click: Run, Run Interactive, Build Image..., Tag..., Push..., Pull, Copy Full Tag, Inspect, Remove, Show / Hide dangling images, Prune

Registries Connect Docker Hub, Azure Container Registry or another registry; pull, push, delete, deploy to Azure

Networks / Volumes Create, Inspect, Remove and Prune; built-in networks can be hidden

Contexts Switch between Docker contexts (for example a remote engine) with *Use*

Configure Explorer... On each view: choose how items are grouped, labelled and sorted

Containers: Prune System... Remove unused containers, images, networks and volumes in one go

DOCKER COMPOSE

Compose Up Start every service in the Compose file

Compose Up - Select Services Choose which services to start

Compose Down / Compose Down - Select Services Stop all

Compose Restart Restart the services

Compose Groups After Compose Up, the Containers view groups Restart, Down for the whole group

containers.composeBuild Build images first when running Compose

containers.composeDetached Run in the background (default)

containers.orchestratorClient Which Compose engine to use: nerdctl, Finch (default auto)

EXPLORER DISPLAY SETTINGS (CONTAINERS.*)

`containers.containers.groupBy` Default `Compose Project Name`; others include `Image`, `State`, `Status`, `Ports`, `None`

`containers.containers.label / .description / .sortBy` Defaults `FullTag`; `[ContainerName, Status]`; `CreatedTime`

`containers.images.groupBy / .label / .sortBy` Defaults `Repository`, `Tag`, `CreatedTime`

`containers.images.checkForOutdatedImages` Mark images with a newer version online (default `true`)

`containers.networks.groupBy / .label` Defaults `None` and `NetworkName`

`containers.networks.showBuiltInNetworks` List the built-in networks (default `true`)

`containers.volumes.groupBy / .label` Defaults `None` and `VolumeName`

`containers.contexts.showInStatusBar` Show the current Docker context in the status bar (default `false`)

`containers.truncateLongRegistryPaths` Shorten long registry paths in the tree (default `false`; length in `truncateMaxLength`, default 10)

ENGINE, COMMANDS & REGISTRIES

`containers.containerClient` Which engine: Docker, Podman, etc. (default `Docker`)

`containers.containerCommand / containers.composeCommand` Command to run in the container (default `docker`)

`Containers: Choose container runtime...` Command Palette option to select a runtime

`containers.environment` Extra environment variables for every container (default `{}`)

`containers.commands.build` Template for building an image: `{containerCommand} build`

`containers.commands.run` Template for Run: `{containerCommand} run --rm -it $@`

`containers.commands.runInteractive` Template for Run Interactive: `... run --rm -it $@`

`containers.commands.attach / .logs` Templates for Attach: `logs --tail 1000`

`containers.imageBuildContextPath` Folder used as the build context (default `.`)

`containers.promptForRegistryWhenPushingImages` Ask whether to push to a registry (default `false`)

`containers.showRemoteWorkspaceWarning` Warn when running in a remote workspace (default `true`)

TASKS: DOCKER-BUILD, DOCKER-RUN, DOCKER-COMPOSE

`docker-build → dockerBuild` `context` (required), `dockerfile`, `tag`, `target`, `buildArgs`, `labels`, `platform`, `pull`, `customOptions`

`docker-run → dockerRun` `image`, `containerName`, `command`, `entrypoint`, `env`, `envFiles`, `network`, `remove` (default `false`), `customOptions`

`dockerRun.ports` `[{ "containerPort": 8000, "hostPort": 8000, "protocol": "tcp" }]`; `portsPublishAll` defaults to `true`

`dockerRun.volumes` `[{ "localPath": ..., "containerPath": ..., "permissions": "ro" }]` (`rw` or `ro`)

`platform: node | python | netCore` Optional language helpers; for Python, `python.file` or `python.module`, `wait` (default `true`), `debugPort` (default 5678)

`docker-compose → dockerCompose` `up` (`detached`, `build`, `services`, `profiles`, `scale`), `down` (`removeImages`, `removeVolumes`, `files`, `envFile`, `projectName`)

Use them as A `preLaunchTask` in `launch.json`, so debugging builds and starts the container first (see the Tasks and Debugging sheets)

DEBUGGING INSIDE A CONTAINER

`Containers: Initialize for container debugging` Command to initialize the debugger configuration

`Debugger type docker` Starts your app in a container and attaches the debugger

`Debugger type python-container` The Python version: debugpy

`Debug in Container` The label of the launch configuration that runs in the container

`Python port` The Python task's `debugPort` (default 5678) must be the same as the one in the launch configuration

DOCKER CLI YOU'LL MEET (MY KNOWLEDGE)

`docker build -t name .` Build an image from the Dockerfile in this folder

`docker run --rm -d -p 8000:8000 name` Run in the background, remove when stopped, map host port 8000 to container port 8000

`docker run --rm -it name sh` Run interactively with a shell

`docker ps / docker ps -a` List running containers / all containers

`docker logs -f id` Follow a container's log

`docker exec -it id sh` Open a shell in a running container

`docker stop id / docker rm id` Stop a container / delete it

`docker images / docker rmi name` List images / remove one

`docker system prune` Remove unused data (asks first)

`docker compose up -d --build` Build and start Compose services in the background

`docker compose down / docker compose logs -f` Stop and remove / follow all logs

A REASONABLE STARTER SET (MY SUGGESTIONS, NOT D

`"containers.containers.groupBy": "Compose Project Nam`

`"containers.contexts.showInStatusBar": true` See which

`"containers.images.checkForOutdatedImages": false` If

`"docker.languageserver.diagnostics.instructionCasing"`

A `.dockerignore` file Always add one: list `node_module` small and free of secrets

THE CONTAINER EXPLORER IS EMPTY OR SAYS DOCKER WAS NOT FOUND

1) Is Docker (or Docker Desktop) installed and *running*? 2) Does `docker version` work in a terminal? 3) Run *Containers: Choose container runtime...* in case the extension picked a different engine. 4) If you run Docker inside WSL, see `dev.containers.executeInWSL` on the Remote and Containers sheet. 5) Check `containers.containerCommand` and `containers.containerClient` are not set to something that does not exist.

DEV CONTAINERS AND CONTAINER TOOLS ARE TWO FEATURES

Container Tools (this sheet) manages containers, images and Compose projects *from* VS Code. **Dev Containers** opens a whole VS Code window *inside* a container, using a `devcontainer.json`. They share Docker, so both need it running, but their settings and commands are separate.