

VS Code Remote and Containers Cheat Sheet

SIDEBAR Programming

Sidebar

Programming · VS Code Remote and Containers Cheat Sheet

Working inside WSL, dev containers and other remote environments: the settings to search for, the commands, the `devcontainer.json` file and port forwarding. Sheet 14 in the VS Code series. Checked against the WSL (0.104.3), Dev Containers (0.469.0) and Containers (2.5.2) extensions installed on this machine; Remote-SSH and Remote Tunnels are separate extensions and are not covered.

WORKING REMOTELY: THE BASICS

Remote indicator (bottom-left) The green button in the status bar: click it for the remote menu, or to see where you are connected

code . Run in a WSL terminal to open that folder in VS Code connected to WSL

WSL: Connect to WSL Command Palette: open a window running inside your default WSL distribution

WSL: Open Folder in WSL... Choose a folder inside WSL to open

WSL: Reopen Folder in WSL / in Windows Switch the current folder between WSL and Windows

Dev Containers: Reopen in Container Reopen the folder inside a container built from its dev container files

Dev Containers: Reopen Folder Locally Leave the container and go back to a normal window

Extensions run where you connect Some install in the remote side: look for *Install in WSL / Container* on an extension

DEVCONTAINER.JSON (THE FILE)

.devcontainer/devcontainer.json Where the container definition lives (also allowed: **.devcontainer.json** in the folder root)

name The label shown for the container

image A ready-made image to use, e.g. `mcr.microsoft.com/devcontainers/python:3`

build.dockerfile Build from your own Dockerfile instead of using **image**

dockerComposeFile / service Use a Compose file and say which service is your workspace

features Add ready-made tools (Node, Git, Docker CLI...) to the container

forwardPorts Ports to forward to your machine, e.g. `[8000]`

postCreateCommand A command run once after the container is created, e.g. `pip install -r requirements.txt`

customizations.vscode.extensions Extensions installed automatically inside the container

customizations.vscode.settings Settings applied inside the container

remoteUser The user VS Code runs as inside the container

containerEnv / mounts / runArgs Environment variables, extra mounts and extra `docker run` options

WSL SETTINGS (REMOTE.WSL.*)

remote.WSL.fileWatcher.polling Poll for file changes instead of using notifications if changes are missed

remote.WSL.fileWatcher.pollingInterval Milliseconds between polls (default 500)

remote.WSL.useShellEnvironment Use your shell's environment for the VS Code session

remote.WSL.downloadInBackground Download the VS Code server in the background

remote.WSL.debug Extra logging for connection problems (default `false`)

remote.WSL1.connectionMethod / remote.WSL2.connectionMethod Choose how to connect (default empty)

terminal.integrated.useWslProfiles List WSL distributions as terminal profiles (default empty)

WSL: Show Log Command Palette: open the log when WSL will not connect

DEV CONTAINERS COMMANDS

Dev Containers: Add Dev Container Configuration Files... Create a `.devcontainer.json` template

Dev Containers: Rebuild Container Rebuild after you change `devcontainer.json`

Dev Containers: Rebuild Without Cache and Reopen in Container A clean rebuild, reopening the folder if stuck

Dev Containers: Attach to Running Container... Open VS Code inside a container

Dev Containers: Open Folder in Container... Pick a folder and open it in a container

Dev Containers: Clone Repository in Container Volume... Clone into a Docker volume instead of a mounted folder

Dev Containers: Show Container Log Read the build and start-up output

Dev Containers: Clean Up Dev Containers... Remove old dev containers

Dev Containers: Configure Container Features... Add tools to the container from the marketplace

Dev Containers: Try a Dev Container Sample... Open a ready-made sample project

DEV CONTAINERS SETTINGS (DEV.CONTAINERS.*)

<code>dev.containers.defaultExtensions</code>	Extension IDs installed in every dev container (default <code>[]</code>)
<code>dev.containers.defaultFeatures</code>	Dev container features added to every container (default <code>{}</code>)
<code>dev.containers.dockerPath</code>	The Docker program to run (default <code>docker</code>)
<code>dev.containers.dockerComposePath</code>	The Compose program to run (default <code>docker-compose</code>)
<code>dev.containers.workspaceMountConsistency</code>	<code>cached</code> (default), <code>consistent</code> , <code>delegated</code> : file sharing mode (mostly macOS)
<code>dev.containers.copyGitConfig</code>	Copy your Git name and email into the container (default <code>true</code>)
<code>dev.containers.gitCredentialHelperConfigLocation</code>	<code>global</code> (default), <code>system</code> , <code>none</code> : where the Git credential helper is set up
<code>dev.containers.cacheVolume</code>	Use a volume to speed up repeated builds (default <code>true</code>)
<code>dev.containers.gpuAvailability</code>	<code>detect</code> (default), <code>all</code> , <code>none</code> : pass GPUs to the container
<code>dev.containers.executeInWSL</code> / <code>executeInWSLDistro</code>	Run Docker through WSL instead of Windows (default <code>false</code>)
<code>dev.containers.logLevel</code>	<code>debug</code> (default), <code>info</code> , <code>trace</code>
<code>dotfiles.repository</code> / <code>dotfiles.installCommand</code>	A Git repository of your dotfiles, cloned into each container, and the command that installs them
<code>dotfiles.targetPath</code>	Where the dotfiles go inside the container (default <code>~/dotfiles</code>)

EXTENSIONS & THE REMOTE SIDE

<code>remote.extensionKind</code>	Force an extension to run on the UI (local) or workspace (remote) side
<code>remote.downloadExtensionsLocally</code>	Download extensions on your machine and send them across (default <code>false</code>)
<code>remote.serverDownloadBaseUrl</code>	Where the VS Code server is downloaded from (default the standard update site)
<code>Install in WSL / Container</code>	Button on an extension in the Extensions view when connected
<code>Disable (Workspace)</code>	Turn an extension off for just this remote folder

PORTS (REMOTE.*)

<code>remote.autoForwardPorts</code>	Forward ports when a program starts listening on them (default <code>true</code>)
<code>remote.autoForwardPortsSource</code>	<code>process</code> (default), <code>output</code> , <code>hybrid</code> : how VS Code finds the port
<code>remote.forwardOnOpen</code>	Forward a port when you open its URL (default <code>true</code>)
<code>remote.restoreForwardedPorts</code>	Re-create forwarded ports after a reload (default <code>true</code>)
<code>remote.localPortHost</code>	<code>localhost</code> (default) or <code>allInterfaces</code> : who on your machine listens on forwarded ports
<code>remote.portsAttributes</code>	Per-port options, e.g. <code>{ "8000": { "label": "FastAPI", "onAutoForward": "notify" } }</code>
<code>onAutoForward values</code>	<code>notify</code> , <code>openBrowser</code> , <code>openBrowserOnce</code> , <code>openPreview</code> , <code>ignore</code>
<code>remote.otherPortsAttributes</code>	Options for every port not listed in <code>remote.portsAttributes</code> (default <code>notify</code>)
<code>Ports view</code>	Panel tab listing forwarded ports, with an address to click and a button to toggle

CONTAINERS (DOCKER) EXTENSION

<code>containers.containerClient</code>	Which engine to use: Docker, Podman, nerdctl, Finch, etc.
<code>containers.containerCommand</code> / <code>composeCommand</code>	Override the program names (default <code>docker</code> / <code>docker-compose</code>)
<code>containers.containers.groupBy</code>	How the Containers view groups items (default <code>Repository</code>)
<code>containers.containers.label</code> / <code>sortBy</code>	Label (default <code>FullTag</code>) and sort order (default <code>asc</code>)
<code>containers.images.groupBy</code> / <code>label</code>	Images grouped by <code>Repository</code> and label (default <code>Repository</code>)
<code>containers.images.checkForOutdatedImages</code>	Mark images that have a newer version (default <code>true</code>)
<code>containers.composeBuild</code> / <code>composeDetached</code>	Compose Up builds first / runs in detached mode (default <code>true</code>)
<code>containers.promptForRegistryWhenPushingImages</code>	Ask which registry to push to (default <code>docker.io</code>)
<code>containers.showRemoteWorkspaceWarning</code>	Warn when you start containers from a remote workspace (default <code>true</code>)
<code>containers.commands.build</code> / <code>run</code> / <code>attach</code> / <code>logs</code>	Templates for the commands to add your own options
<code>docker.languageserver.diagnostics.*</code>	Dockerfile checks, each <code>ignore</code> , <code>warn</code> , or <code>error</code>

A REASONABLE STARTER SET (MY SUGGESTIONS, NOT DEFAULTS)

Keep projects inside WSL Under `/home/you/...` , not `/mnt/c/...` : much faster file access

`"dev.containers.defaultExtensions": [ "ms-python.python" ]` Python tooling in every dev container (use your own list)

`"remote.portsAttributes": { "8000": { "label": "API" } }` A friendly name for the port your server uses

`"remote.WSL.fileWatcher.polling": true` Only if file changes in WSL are not noticed

`"dev.containers.logLevel": "info"` Less noise in the container log

SLOW OR MISSING FILE CHANGES IN WSL? CHECK WHERE THE FILES LIVE

A project stored on the Windows drive (under `/mnt/c`) is much slower to read from inside WSL than one stored in the Linux file system (under `/home`), and file change notifications across that boundary are less reliable. Keep the project in the WSL file system when you work from WSL; only if changes are still missed, turn on `remote.WSL.fileWatcher.polling`.

CONTAINER CHANGES DIDN'T APPLY? REBUILD

Editing `devcontainer.json` or the `Dockerfile` does nothing to a running container. Run *Dev Containers: Rebuild Container*, or *Rebuild Without Cache and Reopen in Container* if the old layers keep being reused. Anything you installed by hand inside the container disappears on rebuild, so put it in `postCreateCommand` or `features`.