Docker Toolbox

Complete Reference Guide

71 Development Tools via Docker

Run popular development tools without installing them locally

Generated from tools.yaml

1 Table of Contents

Contents

2 Introduction

Docker Toolbox provides ready-to-use Docker commands for development tools across multiple categories. This reference guide contains all available tools with usage examples.

2.1 Benefits

- **No local installation** required (just Docker)
- **Consistent behavior** across Linux, macOS, and Windows
- **Isolated environments** prevent conflicts
- **Easy to try** new tools without commitment
- **Clean uninstall** - just remove the container

3 Categories Overview

Category	Description	Tools
Terminal Tools	Command-line utilities and terminal enhancements	11
Programming Languages	Language runtimes and interpreters	5
Development Environments & IDEs	Complete development environments and IDEs	7
Static Site Generators	Tools for building static websites	3
Build & Task Runners	Task automation and build tools	2
Testing Tools	Testing frameworks and tools	1
Databases	Database servers and clients	6
Monitoring & Visualization	Monitoring, metrics, and dashboards	3
Message Brokers & IoT	MQTT, message queues, and IoT platforms	1
DevOps & Cloud CLI	Cloud CLIs and infrastructure tools	8
Code Quality & Linting	Code formatters and linters	5
Media & Documents	Media processing and document conversion	6
Networking & Security	Network tools and security scanners	4
Web Servers & Security	Web servers and security tools	2
API Development	API testing and development tools	3
Git Tools	Git and version control tools	2
AI & Machine Learning	AI models and machine learning tools	2

4 Tools Reference

4.1 Terminal Tools

4.1.1 Tmux

Terminal multiplexer for managing multiple sessions

Docker Image: `alpine`

Usage:

```
# Basic usage
docker run --rm -it -v ${PWD}:/workspace -v ~/.tmux.conf:/root/.tmux.conf -w /workspace
alpine sh -c "apk add --no-cache tmux bash git && tmux"
```

Aliases:

Bash/Zsh: `dttmux`

Note: Container and tmux sessions are deleted when you exit. Mount your own `.tmux.conf` for custom settings.

4.1.2 Htop

Interactive process viewer

Docker Image: `alpine`

Usage:

```
# Basic usage
docker run --rm -it --pid=host alpine sh -c "apk add --no-cache htop && htop"
```

Aliases:

Bash/Zsh: `dthtop`

4.1.3 Lazygit

Simple terminal UI for git commands

Docker Image: `lazyteam/lazygit`

Usage:

```
# Basic usage
docker run --rm -it -v ${PWD}:/repo -w /repo lazyteam/lazygit
```

Aliases:

Bash/Zsh: `dtlazygit`

4.1.4 Lazydocker

Simple terminal UI for Docker management

Docker Image: `lazyteam/lazydocker`

Usage:

```
# Basic usage
docker run --rm -it -v /var/run/docker.sock:/var/run/docker.sock lazyteam/lazydocker
```

Aliases:

Bash/Zsh: `dtlazydocker`

4.1.5 Ripgrep

Lightning-fast search tool (better than grep)

Docker Image: `alpine`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/data -w /data alpine sh -c "apk add --no-cache ripgrep > /dev/
null 2>&1 && rg $args"
```

Aliases:

Bash/Zsh: `dtrg`

4.1.6 Fd

Fast and user-friendly alternative to find

Docker Image: `alpine`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/data -w /data alpine sh -c "apk add --no-cache fd > /dev/null
2>&1 && fd $args"
```

Aliases:

Bash/Zsh: `dtfd`

4.1.7 Bat

Cat clone with syntax highlighting

Docker Image: `alpine`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/data -w /data alpine sh -c "apk add --no-cache bat > /dev/null
2>&1 && bat $args"
```

Aliases:

Bash/Zsh: `dtbat`

4.1.8 Jq

Lightweight JSON processor

Docker Image: `ghcr.io/jqlang/jq`

Usage:

```bash
# Basic usage
docker run --rm -i ghcr.io/jqlang/jq $args
```

Aliases:

Bash/Zsh: `dtjq`

4.1.9 Yq

YAML/JSON/XML processor (like jq for YAML)

Docker Image: `mikefarah/yq`

Usage:

```bash
# Basic usage
docker run --rm -v ${PWD}:/workdir mikefarah/yq $args
```

Aliases:

Bash/Zsh: `dtyq`

4.2 Programming Languages

4.2.1 Python

Python 3.12 interpreter and runtime

Docker Image: `python:3.12`

Usage:

```
# Basic usage
docker run --rm -it -v ${PWD}:/app -w /app python:3.12 python $args
```

```
# Ipython
docker run --rm -it -v ${PWD}:/app -w /app python:3.12 sh -c "pip install -q ipython &&
ipython"
```

Aliases:

Bash/Zsh: `dtpython`, `dtipython`

Examples:

- *Run Python script*

  ```
  dtpython script.py
  ```

- *Interactive Python shell*

  ```
  dtpython
  ```

- *IPython shell*

  ```
  dtipython
  ```

4.2.2 Node

Node.js 22 runtime

Docker Image: `node:22`

Usage:

```
# Basic usage
docker run --rm -it -v ${PWD}:/app -w /app node:22 node $args
```

```
# Npm
docker run --rm -v ${PWD}:/app -w /app node:22 npm $args
```

```
# Npx
docker run --rm -v ${PWD}:/app -w /app node:22 npx $args
```

```
# Yarn
docker run --rm -v ${PWD}:/app -w /app node:22 yarn $args
```

Aliases:

Bash/Zsh: `dtnode`, `dtnpm`, `dtnpx`

4.2.3 Go

Go language compiler and runtime

Docker Image: `golang:1.22`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/app -w /app golang:1.22 go $args
```

Aliases:

Bash/Zsh: `dtgo`

4.2.4 Ruby

Ruby interpreter

Docker Image: `ruby:3.3`

Usage:

```
# Basic usage
docker run --rm -it -v ${PWD}:/app -w /app ruby:3.3 ruby $args
```

```
# Irb
docker run --rm -it -v ${PWD}:/app -w /app ruby:3.3 irb
```

```
# Bundle
docker run --rm -v ${PWD}:/app -w /app ruby:3.3 bundle $args
```

Aliases:

Bash/Zsh: `dtruby`, `dtirb`, `dtbundle`

4.2.5 Cargo

Rust package manager and build tool

Docker Image: `rust:latest`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/app -v cargo-cache:/usr/local/cargo -w /app rust:latest cargo
$args
```

```
# Rustc
docker run --rm -v ${PWD}:/app -w /app rust:latest rustc $args
```

Aliases:

Bash/Zsh: `dtcargo`, `dtrustc`

Note: The cargo-cache volume persists downloaded dependencies between runs.

4.3 Development Environments & IDEs

4.3.1 Jupyter

Jupyter Notebook for data science and analysis

Docker Image: `jupyter/base-notebook`

Usage:

```
# Basic usage
docker run --rm -p 8888:8888 -v ${PWD}:/home/jovyan/work jupyter/base-notebook
```

```
# Jupyterlab
docker run --rm -p 8888:8888 -v ${PWD}:/home/jovyan/work jupyter/datascience-notebook
```

```
# Scipy
docker run --rm -p 8888:8888 -v ${PWD}:/home/jovyan/work jupyter/scipy-notebook
```

Aliases:

Bash/Zsh: `dtjupyter`, `dtjupyterlab`, `dtjupyterscipy`

Note: Access at http://localhost:8888. Copy the token from the logs.

4.3.2 Vscode

VS Code Server in browser

Docker Image: `codercom/code-server`

Usage:

```
# Basic usage
docker run --rm -p 8080:8080 -v ${PWD}:/home/coder/project codercom/code-server --auth none
```

Aliases:

Bash/Zsh: `dtvscode`

Note: Access at http://localhost:8080. No authentication by default.

4.3.3 Rstudio

RStudio IDE for R programming

Docker Image: `rocker/rstudio`

Usage:

```
# Basic usage
docker run --rm -p 8787:8787 -e PASSWORD=rstudio -v ${PWD}:/home/rstudio rocker/rstudio
```

```
# Tidyverse
docker run --rm -p 8787:8787 -e PASSWORD=rstudio -v ${PWD}:/home/rstudio rocker/tidyverse
```

Aliases:

Bash/Zsh: `dtrstudio`, `dtrstudiotidy`

Note: Access at http://localhost:8787. Login: rstudio / rstudio

4.3.4 Vert

Web-based terminal and development environment

Docker Image: `ghcr.io/vert-sh/vert:latest`

Aliases:

Bash/Zsh: `dtvert`, `dtvertstop`

Note: Supports multiple instances on different ports. Container persists between restarts.

4.3.5 Webtop

Full Linux desktop environment in browser (no X11 needed)

Docker Image: `lscr.io/linuxserver/webtop:ubuntu-mate`

Usage:

```
# Basic usage
docker run -d --name=webtop -p 3000:3000 -p 3001:3001 -v webtop-config:/config --shm-
size="1gb" --restart unless-stopped lscr.io/linuxserver/webtop:ubuntu-mate
```

```
# Xfce
docker run -d --name=webtop -p 3000:3000 -p 3001:3001 -v webtop-config:/config --shm-
size="1gb" --restart unless-stopped lscr.io/linuxserver/webtop:ubuntu-xfce
```

```
# Start
docker start webtop
```

```
# Stop
docker stop webtop
```

```
# Logs
docker logs webtop -f
```

Aliases:

Bash/Zsh: `dtwebtop`, `dtwebtopxfce`, `dtwebtopstart`

Note: Access at http://localhost:3000. Full Ubuntu desktop with Firefox, terminal, and file manager. Default password: abc

4.3.6 Nodered

Flow-based programming for IoT and automation

Docker Image: `nodered/node-red`

Usage:

```
# Basic usage
docker run --rm -p 1880:1880 -v nodered-data:/data nodered/node-red
```

```
# Start
docker start nodered
```

```
# Stop
docker stop nodered
```

```
# Logs
docker logs nodered -f
```

Aliases:

Bash/Zsh: `dtnodered`, `dtnoderedstart`, `dtnoderedstop`

Note: Access at http://localhost:1880. Flows stored in nodered-data volume.

4.3.7 N8N

Workflow automation (Zapier/Make alternative)

Docker Image: `n8nio/n8n`

Usage:

```
# Basic usage
docker run --rm -p 5678:5678 -v n8n-data:/home/node/.n8n n8nio/n8n
```

```
# Start
docker start n8n
```

```
# Stop
docker stop n8n
```

```
# Logs
docker logs n8n -f
```

Aliases:

Bash/Zsh: `dtn8n`, `dtn8nstart`, `dtn8nstop`

Note: Access at http://localhost:5678. Supports 200+ integrations.

4.4 AI & Machine Learning

4.4.1 Ollama

Run large language models locally (Llama, Mistral, etc)

Docker Image: `ollama/ollama`

Usage:

```
# Start
docker run -d --restart unless-stopped -p 11434:11434 -v ollama-data:/root/.ollama --name
ollama ollama/ollama
```

```
# Stop
docker stop ollama
```

```
# Exec
docker exec -it ollama ollama $args
```

```
# Run
docker exec -it ollama ollama run $args
```

```
# Pull
docker exec -it ollama ollama pull $args
```

```
# List
docker exec -it ollama ollama list
```

Aliases:

Bash/Zsh: `dtollama`, `dtollamastart`, `dtollamastop`

Note: Start server first, then pull models. API at http://localhost:11434

4.4.2 Openwebui

ChatGPT-style web interface for Ollama

Docker Image: `ghcr.io/open-webui/open-webui`

Usage:

```
# Basic usage
docker run -d --restart unless-stopped -p 3000:8080 -v open-webui-data:/app/backend/data --
add-host=host.docker.internal:host-gateway --name open-webui ghcr.io/open-webui/open-webui
```

```
# Start
docker start open-webui
```

```
# Stop
docker stop open-webui
```

```
# Logs
docker logs open-webui -f
```

Aliases:

Bash/Zsh: `dtopenwebui`, `dtopenwebuistart`, `dtopenwebuistop`

Note: Access at http://localhost:3000. Requires Ollama running. First user to register becomes admin.

4.5 Databases

4.5.1 Postgres

PostgreSQL relational database

Docker Image: `postgres:16`

Usage:

```bash
# Basic usage
docker run --rm -p 5432:5432 -e POSTGRES_PASSWORD=secret postgres:16
```

```bash
# Psql
docker run --rm -it postgres:16 psql $args
```

Aliases:

Bash/Zsh: `dtpostgres`, `dtpsql`

4.5.2 Mysql

MySQL relational database

Docker Image: `mysql:8`

Usage:

```bash
# Basic usage
docker run --rm -p 3306:3306 -e MYSQL_ROOT_PASSWORD=secret mysql:8
```

```bash
# Client
docker run --rm -it mysql:8 mysql $args
```

Aliases:

Bash/Zsh: `dtmysql`, `dtmysqlclient`

4.5.3 Redis

Redis in-memory data store

Docker Image: `redis:7-alpine`

Usage:

```bash
# Basic usage
docker run --rm -p 6379:6379 redis:7-alpine
```

```bash
# Cli
docker run --rm -it redis:7-alpine redis-cli $args
```

Aliases:

Bash/Zsh: `dtredis`, `dtrediscli`

4.5.4 Mongo

MongoDB NoSQL document database

Docker Image: `mongo:7`

Usage:

```
# Basic usage
docker run --rm -p 27017:27017 mongo:7
```

```
# Mongosh
docker run --rm -it mongo:7 mongosh $args
```

Aliases:

Bash/Zsh: `dtmongo`, `dtmongosh`

4.5.5 Influxdb

Time-series database for IoT and metrics

Docker Image: `influxdb:2`

Usage:

```
# Basic usage
docker run --rm -p 8086:8086 -v influxdb-data:/var/lib/influxdb2 influxdb:2
```

```
# Start
docker start influxdb
```

```
# Stop
docker stop influxdb
```

```
# Cli
docker exec -it influxdb influx
```

Aliases:

Bash/Zsh: `dtinfluxdb`, `dtinfluxstart`, `dtinfluxstop`

Note: Access UI at http://localhost:8086. Perfect for storing sensor data.

4.5.6 Nocodb

Turn any database into a smart spreadsheet (Airtable alternative)

Docker Image: `nocodb/nocodb`

Usage:

```
# Basic usage
docker run --rm -p 8080:8080 -v nocodb-data:/usr/app/data nocodb/nocodb
```

```
# Start
docker start nocodb
```

```
# Stop
docker stop nocodb
```

```
# Logs
docker logs nocodb -f
```

Aliases:

Bash/Zsh: `dtnocodb`, `dtnocodbstart`, `dtnocodbstop`

Note: Access at http://localhost:8080. Works with PostgreSQL, MySQL, SQLite.

4.6 Monitoring & Visualization

4.6.1 Grafana

Beautiful dashboards for metrics and time-series data

Docker Image: `grafana/grafana`

Usage:

```
# Basic usage
docker run --rm -p 3000:3000 -v grafana-data:/var/lib/grafana grafana/grafana
```

```
# Start
docker start grafana
```

```
# Stop
docker stop grafana
```

```
# Logs
docker logs grafana -f
```

Aliases:

Bash/Zsh: `dtgrafana`, `dtgrafanastart`, `dtgrafanastop`

Note: Access at http://localhost:3000. Default login: admin/admin

4.6.2 Uptime-Kuma

Self-hosted uptime monitoring with notifications

Docker Image: `louislam/uptime-kuma`

Usage:

```
# Basic usage
docker run --rm -p 3001:3001 -v uptime-kuma-data:/app/data louislam/uptime-kuma
```

```
# Start
docker start uptime-kuma
```

```
# Stop
docker stop uptime-kuma
```

```
# Logs
docker logs uptime-kuma -f
```

Aliases:

Bash/Zsh: `dtuptime`, `dtuptimestart`, `dtuptimestop`

Note: Access at http://localhost:3001. Supports 50+ notification services.

4.6.3 Dozzle

Real-time Docker container log viewer with web UI

Docker Image: `amir20/dozzle`

Usage:

```
# Basic usage
docker run --rm -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock:ro amir20/dozzle
```

```
# Start
docker start dozzle
```

```
# Stop
docker stop dozzle
```

Aliases:

Bash/Zsh: `dtdozzle`, `dtdozzlestart`, `dtdozzlestop`

Note: Access at http://localhost:8080. Zero configuration needed!

4.7 DevOps & Cloud CLI

4.7.1 Localstack

Local AWS cloud stack for testing (S3, Lambda, DynamoDB, etc)

Docker Image: `localstack/localstack`

Usage:

```
# Basic usage
docker run --rm -p 4566:4566 -v localstack-data:/var/lib/localstack localstack/localstack
```

```
# Start
docker start localstack
```

```
# Stop
docker stop localstack
```

```
# Logs
docker logs localstack -f
```

Aliases:

Bash/Zsh: `dtlocalstack`, `dtlocalstackstart`, `dtlocalstackstop`

Note: All services at http://localhost:4566. Use AWS CLI with –endpoint-url=http://localhost:4566

4.7.2 Vault

Secrets management by HashiCorp

Docker Image: `hashicorp/vault`

Usage:

```
# Basic usage
docker run --rm -p 8200:8200 -e VAULT_DEV_ROOT_TOKEN_ID=myroot hashicorp/vault
```

```
# Start
docker start vault
```

```
# Stop
docker stop vault
```

```
# Cli
docker exec -it vault vault $args
```

Aliases:

Bash/Zsh: `dtvault`, `dtvaultstart`, `dtvaultstop`

Note: Access at http://localhost:8200. Dev mode token: myroot. NOT for production!

4.8 Web Servers & Security

4.8.1 Caddy

Modern web server with automatic HTTPS

Docker Image: `caddy`

Usage:

```
# Basic usage
docker run --rm -p 80:80 -p 443:443 -v ${PWD}:/usr/share/caddy caddy
```

```
# Start
docker start caddy
```

```
# Stop
docker stop caddy
```

```
# Reload
docker exec -w /etc/caddy caddy caddy reload
```

Aliases:

Bash/Zsh: `dtcaddy`, `dtcaddystart`, `dtcaddystop`

Note: Automatic SSL certificates from Let's Encrypt. Zero-downtime reloads.

4.8.2 Vaultwarden

Lightweight Bitwarden password manager server

Docker Image: `vaultwarden/server`

Usage:

```
# Basic usage
docker run --rm -p 8080:80 -v vaultwarden-data:/data vaultwarden/server
```

```
# Start
docker start vaultwarden
```

```
# Stop
docker stop vaultwarden
```

```
# Logs
docker logs vaultwarden -f
```

Aliases:

Bash/Zsh: `dtvaultwarden`, `dtvaultwardenstart`, `dtvaultwardenstop`

Note: Access at http://localhost:8080. Compatible with Bitwarden clients. Use HTTPS in production!

4.9 Message Brokers & IoT

4.9.1 Mosquitto

Lightweight MQTT broker for IoT

Docker Image: `eclipse-mosquitto`

Usage:

```
# Basic usage
docker run --rm -p 1883:1883 -p 9001:9001 eclipse-mosquitto
```

```
# Sub
docker run --rm eclipse-mosquitto mosquitto_sub -h host.docker.internal $args
```

```
# Pub
docker run --rm eclipse-mosquitto mosquitto_pub -h host.docker.internal $args
```

```
# Passwd
docker run --rm -v ${PWD}:/mosquitto/config eclipse-mosquitto mosquitto_passwd $args
```

Aliases:

Bash/Zsh: `dtmosquitto`, `dtmqttsub`, `dtmqttpub`

Note: Port 1883 for MQTT, 9001 for WebSockets. Use host.docker.internal to connect.

4.10 Static Site Generators

4.10.1 Jekyll

Ruby-based static site generator, popular for GitHub Pages

Docker Image: `jekyll/jekyll`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/srv/jekyll jekyll/jekyll jekyll $args
```

```
# Serve
docker run --rm -v ${PWD}:/srv/jekyll -p 4000:4000 jekyll/jekyll jekyll serve --watch --
force_polling
```

```
# Simple
docker run --rm -v ${PWD}:/site -p 4000:4000 bretfisher/jekyll-serve
```

```
# Build
docker run --rm -v ${PWD}:/srv/jekyll jekyll/jekyll jekyll build
```

Aliases:

Bash/Zsh: `dtjekyll`, `dtjekyllserve`, `dtjekyllsimple`

Note: The bretfisher/jekyll-serve image auto-runs bundle install and is useful for complex setups.

4.10.2 Hugo

Fast static site generator written in Go

Docker Image: `klakegg/hugo`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/src klakegg/hugo $args
```

```
# Serve
docker run --rm -v ${PWD}:/src -p 1313:1313 klakegg/hugo server --bind 0.0.0.0
```

```
# New
docker run --rm -v ${PWD}:/src klakegg/hugo new site mysite
```

Aliases:

Bash/Zsh: `dthugo`, `dthugoserve`

Note: Access at http://localhost:1313

4.10.3 Mkdocs

Python documentation generator with beautiful Material theme

Docker Image: `squidfunk/mkdocs-material`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/docs squidfunk/mkdocs-material $args
```

```
# Serve
docker run --rm -v ${PWD}:/docs -p 8000:8000 squidfunk/mkdocs-material
```

```
# Build
docker run --rm -v ${PWD}:/docs squidfunk/mkdocs-material build
```

```
# New
docker run --rm -v ${PWD}:/docs squidfunk/mkdocs-material new .
```

Aliases:

Bash/Zsh: `dtmkdocs`, `dtmkdocsserve`

Note: Access at http://localhost:8000

4.11 Build & Task Runners

4.11.1 Just

Command runner for executing project-specific tasks and recipes

Docker Image: `alpine`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/workdir -w /workdir alpine sh -c "wget -q https://github.com/
casey/just/releases/download/1.16.0/just-1.16.0-x86_64-unknown-linux-musl.tar.gz -O- | tar -
xz -C /usr/local/bin && just $args"
```

Aliases:

Bash/Zsh: `dtjust`

Note: Binary download adds startup time. For frequent use, consider installing natively.

4.11.2 Make

GNU Make for traditional Makefile execution

Docker Image: `alpine`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/workdir -w /workdir alpine sh -c "apk add --no-cache make > /dev/
null 2>&1 && make $args"
```

Aliases:

Bash/Zsh: `dtmake`

Note: Supports parallel execution with -j flag.

4.12 Testing Tools

4.12.1 Playwright

Browser automation and E2E testing framework

Docker Image: `mcr.microsoft.com/playwright:latest`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/work -w /work -it mcr.microsoft.com/playwright:latest /bin/bash
```

```
# Test
docker run --rm -v ${PWD}:/work -w /work mcr.microsoft.com/playwright:latest npx playwright
test
```

Aliases:

Bash/Zsh: `dtplaywright`, `dtplaywrighttest`

Note: Includes Chromium, Firefox, and WebKit browsers.

4.13 Code Quality & Linting

4.13.1 Prettier

Opinionated code formatter for multiple languages

Docker Image: `tmknom/prettier`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/work tmknom/prettier $args
```

```
# Check
docker run --rm -v ${PWD}:/work tmknom/prettier --check .
```

```
# Write
docker run --rm -v ${PWD}:/work tmknom/prettier --write .
```

Aliases:

Bash/Zsh: `dtprettier`, `dtprettiercheck`, `dtprettierwrite`

4.13.2 Black

Python code formatter

Docker Image: `cytopia/black`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/data cytopia/black $args
```

```
# Check
docker run --rm -v ${PWD}:/data cytopia/black --check .
```

Aliases:

Bash/Zsh: `dtblack`, `dtblackcheck`

4.13.3 Shellcheck

Static analysis tool for shell scripts

Docker Image: `koalaman/shellcheck`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/mnt koalaman/shellcheck $args
```

Aliases:

Bash/Zsh: `dtshellcheck`

4.13.4 Hadolint

Dockerfile linter

Docker Image: `hadolint/hadolint`

Usage:

```
# Basic usage
docker run --rm -i hadolint/hadolint < Dockerfile
```

Aliases:

Bash/Zsh: `dthadolint`

4.13.5 Markdownlint

Markdown linter

Docker Image: `tmknom/markdownlint`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/work tmknom/markdownlint $args
```

Aliases:

Bash/Zsh: `dtmarkdownlint`

4.14 Media & Documents

4.14.1 Pandoc

Universal document converter (Markdown, PDF, DOCX, etc)

Docker Image: `pandoc/latex`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/data pandoc/latex $args
```

Aliases:

Bash/Zsh: `dtpandoc`

Examples:

- *Convert Markdown to PDF*

  ```
  dtpandoc input.md -o output.pdf
  ```

- *Convert DOCX to Markdown*

  ```
  dtpandoc input.docx -o output.md
  ```

4.14.2 Ffmpeg

Media processing tool for video and audio

Docker Image: `jrottenberg/ffmpeg`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/tmp jrottenberg/ffmpeg $args
```

Aliases:

Bash/Zsh: `dtffmpeg`

Examples:

- *Convert video to MP4*

  ```
  dtffmpeg -i input.avi output.mp4
  ```

4.14.3 Imagemagick

Image manipulation and conversion

Docker Image: `dpokidov/imagemagick`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/imgs dpokidov/imagemagick convert $args
```

```
# Identify
docker run --rm -v ${PWD}:/imgs dpokidov/imagemagick identify $args
```

```
# Mogrify
docker run --rm -v ${PWD}:/imgs dpokidov/imagemagick mogrify $args
```

Aliases:

Bash/Zsh: `dtconvert`, `dtidentify`, `dtmogrify`

4.14.4 Yt-Dlp

Download videos from YouTube and other sites

Docker Image: `jauderho/yt-dlp`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/downloads jauderho/yt-dlp $args
```

Aliases:

Bash/Zsh: `dtytdlp`

4.14.5 Typst

Modern markup-based typesetting system

Docker Image: `ghcr.io/typst/typst`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/data ghcr.io/typst/typst compile $args
```

```
# Watch
docker run --rm -v ${PWD}:/data ghcr.io/typst/typst watch $args
```

Aliases:

Bash/Zsh: `dttypst`, `dttypstwatch`

4.14.6 Latex

LaTeX document preparation system

Docker Image: `texlive/texlive`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/workdir -w /workdir texlive/texlive pdflatex $args
```

```
# Latexmk
docker run --rm -v ${PWD}:/workdir -w /workdir texlive/texlive latexmk -pdf $args
```

```
# Xelatex
docker run --rm -v ${PWD}:/workdir -w /workdir texlive/texlive xelatex $args
```

Aliases:

Bash/Zsh: `dtpdflatex`, `dtlatexmk`, `dtxelatex`

4.15 DevOps & Cloud CLI

4.15.1 Awscli

AWS Command Line Interface

Docker Image: `amazon/aws-cli`

Usage:

```
# Basic usage
docker run --rm -v ~/.aws:/root/.aws -v ${PWD}:/aws amazon/aws-cli $args
```

Aliases:

Bash/Zsh: `dtaws`

4.15.2 Azurecli

Azure Command Line Interface

Docker Image: `mcr.microsoft.com/azure-cli`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/work -w /work mcr.microsoft.com/azure-cli az $args
```

Aliases:

Bash/Zsh: `dtaz`

4.15.3 Gcloud

Google Cloud Command Line Interface

Docker Image: `google/cloud-sdk`

Usage:

```
# Basic usage
docker run --rm -v ~/.config/gcloud:/root/.config/gcloud -v ${PWD}:/work -w /work google/
cloud-sdk gcloud $args
```

Aliases:

Bash/Zsh: `dtgcloud`

4.15.4 Terraform

Infrastructure as Code tool

Docker Image: `hashicorp/terraform`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/workspace -w /workspace hashicorp/terraform $args
```

Aliases:

Bash/Zsh: `dtterraform`

4.15.5 Ansible

IT automation and configuration management

Docker Image: `willhallonline/ansible`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/ansible -w /ansible willhallonline/ansible ansible $args
```

```
# Playbook
docker run --rm -v ${PWD}:/ansible -w /ansible willhallonline/ansible ansible-playbook $args
```

Aliases:

Bash/Zsh: `dtansible`, `dtansibleplaybook`

4.15.6 Helm

Kubernetes package manager

Docker Image: `alpine/helm`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/apps -v ~/.kube:/root/.kube alpine/helm $args
```

Aliases:

Bash/Zsh: `dthelm`

4.16 API Development

4.16.1 Swagger-Ui

Interactive API documentation viewer

Docker Image: `swaggerapi/swagger-ui`

Usage:

```
# Basic usage
docker run --rm -p 8080:8080 -e SWAGGER_JSON=/app/swagger.json -v ${PWD}:/app swaggerapi/
swagger-ui
```

Aliases:

Bash/Zsh: `dtswagger`

4.16.2 Httpie

User-friendly HTTP client

Docker Image: `alpine`

Usage:

```
# Basic usage
docker run --rm -it alpine sh -c "apk add --no-cache httpie > /dev/null 2>&1 && http $args"
```

Aliases:

Bash/Zsh: `dthttp`, `dthttpie`

4.16.3 Newman

Postman collection runner

Docker Image: `postman/newman`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/etc/newman postman/newman run $args
```

Aliases:

Bash/Zsh: `dtnewman`

4.17 Git Tools

4.17.1 Git

Git version control

Docker Image: `alpine/git`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/git -w /git alpine/git $args
```

Aliases:

Bash/Zsh: `dtgit`

4.17.2 Gh

GitHub CLI

Docker Image: `ghcr.io/cli/cli`

Usage:

```
# Basic usage
docker run --rm -v ${PWD}:/repo -w /repo -v ~/.config/gh:/root/.config/gh ghcr.io/cli/cli
$args
```

Aliases:

Bash/Zsh: `dtgh`

4.18 Networking & Security

4.18.1 Trivy

Vulnerability scanner for containers and IaC

Docker Image: `aquasec/trivy`

Usage:

```
# Basic usage
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -v ${PWD}:/work aquasec/trivy
$args
```

Aliases:

Bash/Zsh: `dttrivy`

4.18.2 Nmap

Network exploration and security scanning

Docker Image: `instrumentisto/nmap`

Usage:

```
# Basic usage
docker run --rm instrumentisto/nmap $args
```

Aliases:

Bash/Zsh: `dtnmap`

4.18.3 Curl

Command line HTTP client

Docker Image: `curlimages/curl`

Usage:

```
# Basic usage
docker run --rm curlimages/curl $args
```

Aliases:

Bash/Zsh: `dtcurl`

4.18.4 Testssl

SSL/TLS security testing

Docker Image: `drwetter/testssl.sh`

Usage:

```
# Basic usage
docker run --rm drwetter/testssl.sh $args
```

Aliases:

Bash/Zsh: `dttestssl`

4.19 Terminal Tools

4.19.1 Ncdu

Disk usage analyzer with ncurses interface

Docker Image: `alpine`

Usage:

```
# Basic usage
docker run --rm -it -v ${PWD}:/data alpine sh -c "apk add --no-cache ncdu > /dev/null 2>&1
&& ncdu /data"
```

Aliases:

Bash/Zsh: `dtncdu`

4.19.2 Ranger

Console file manager with VI key bindings

Docker Image: `alpine`

Usage:

```
# Basic usage
docker run --rm -it -v ${PWD}:/data alpine sh -c "apk add --no-cache ranger > /dev/null 2>&1
&& ranger /data"
```

Aliases:

Bash/Zsh: `dtranger`

5 Appendix

5.1 Installation

To use Docker Toolbox, you need Docker installed on your system.

Prerequisites:
- Docker 20.10 or later
- Basic command line knowledge

Visit https://github.com/yourusername/docker-toolbox for installation instructions.

5.2 Contributing

Docker Toolbox uses a YAML-based tool management system. To add a new tool:

1. Edit `tools.yaml`
2. Run `python generate.py --all`
3. Test the generated outputs
4. Submit a pull request

5.3 License

MIT License - Free to use, modify, and distribute.