Devbox Nix dev environments

CONCEPTS

devbox.json lists the packages a project needs; devbox.lock pins exact versions. Commit both so everyone gets the same tools.

devbox.json packages, env, scripts
↓ devbox add
devbox.lock pinned versions
↓ devbox shell
Nix store installed packages
↓ PATH
Shell isolated tools

Devbox wraps Nix: you get reproducible packages without writing Nix code. Nix itself is installed for you on first use.

Flags of devbox run must come before the script or command name.

PROJECT SETUP

devbox init
Create devbox.json in this directory

devbox add *<pkg>*
Add a package to the project

devbox add *<pkg>*@*<version>*
Add a specific version

devbox rm *<pkg>*
Remove a package

devbox install
Install all packages without a shell

devbox search *<pkg>*
List matching packages and versions

devbox info *<pkg>*
Show package and plugin info

devbox update
Update all packages within their version tags

devbox update *<pkg>*
Update only the named packages

devbox add *<pkg>* **-p** *<platform>*
Install only on given platforms

SHELL AND RUN

devbox shell
Start a shell with your packages

devbox shell -- *<cmd>*
Run one command in the environment

devbox shell --pure
Isolated shell, almost no inherited env

devbox shell -c *<dir>*
Use the devbox.json in another directory

devbox shell -e *<KEY>*=*<val>*
Set a variable inside the environment

devbox shell --env-file *<file>*
Load variables from a file

devbox run *<script>*
Run a script from devbox.json

devbox run *<cmd>*
Run any command inside the environment

exit
Leave the devbox shell

DEVBOX.JSON

Packages (list form)

```
{
  "packages": ["go@latest",
"nodejs@20"]
}
```

Environment variables

```
"env": {
  "APP_ENV": "dev",
  "BIN": "$PWD/bin"
}
```

"env_from": "*<path>***.env"**
Load variables from a .env file

Init hook

```
"shell": {
  "init_hook": ["echo Welcome",
"export FOO=bar"]
}
```

Scripts

```
"shell": {
  "scripts": {
    "test": "go test ./...",
    "ci": ["go vet ./...", "go
test ./..."]
  }
}
```

init_hook runs every time a shell starts, so keep it fast and idempotent.

SERVICES

devbox services ls
List available services

devbox services up
Start all services with process-compose

devbox services up *<svc>*
Start only the named services

devbox services up -b
Run services in the background

devbox services stop *<svc>*
Stop a service (all if none named)

Packages with a plugin (postgresql, redis, nginx, ...) expose ready-made services; others need a process-compose.yaml next to devbox.json.

GENERATE AND GLOBAL

devbox generate direnv
Write an .envrc that loads the environment

devbox generate dockerfile
Dockerfile that replicates devbox shell

devbox global add *<pkg>*
Install a package for all projects

RECIPES

Start a project

```
devbox init
devbox add python@3.12
postgresql@16
devbox shell
python --version
```

Auto-load with direnv

```
devbox generate direnv
direnv allow
# entering the directory now
loads the environment
```

Scripts and CI

```
# devbox.json: shell.scripts.test
devbox run test
devbox run -e CI=1 test
```

Services in the background

```
devbox services up -b
devbox services ls
devbox services stop
```

Containerize

```
devbox generate dockerfile
docker build -t myapp .
```