Bundler Ruby dependencies

Gemfile your constraints → `install / update` → **Gemfile.lock** exact versions → `install` → **Installed gems** system or vendor/bundle → `exec / setup` → **Your app** load path set

CONCEPTS

Commit Gemfile and Gemfile.lock. The Gemfile holds your constraints, the lockfile pins the exact versions everyone gets.

Bundler 4: bare bundle shows help instead of installing, and flags like --deployment, --path and --without no longer exist. Use bundle config.

PROJECT

bundle init
Write a starter Gemfile here

bundle gem *<name>*
Scaffold a new gem

bundle version
Show the Bundler version

INSTALL & REMOVE

bundle install
Install gems, write Gemfile.lock

bundle add *<gem>*
Add a gem to the Gemfile and install

bundle add *<gem>* **-v "~> 1.2" -g development**
Add with version and group

INSTALL & REMOVE (cont.)

bundle add *<gem>* **--skip-install**
Edit the Gemfile only

bundle remove *<gem>*
Remove a gem from the Gemfile

bundle install --jobs *<n>*
Install in parallel

bundle install --local
Use only gems already on disk

RUN

bundle exec *<cmd>*
Run a command with the bundle's gems

bundle exec rake *<task>*
Run a Rake task in the bundle

bundle exec rspec
Run tests with locked versions

bundle binstubs *<gem>*
Create bin/<exe> wrappers for a gem

bundle check
Are the dependencies satisfied?

UPDATE & OUTDATED

bundle update *<gem>*
Upgrade one gem and what it needs

bundle update --all
Upgrade everything

bundle outdated
List gems with newer versions

bundle update *<gem>* **--conservative**
Keep shared dependencies where they are

bundle update --patch
Prefer patch releases only

INSPECT

bundle list
Every gem in the bundle with version

bundle list --name-only
Just the names

bundle info *<gem>*
Details about one gem

bundle info *<gem>* **--path**
Where a gem is installed

LOCKFILE

bundle lock
Resolve and write the lockfile only

LOCKFILE (cont.)

bundle lock --add-platform *<platform>*
Add a platform to the lockfile

CONFIG & DEPLOY

bundle config set --local *<name> <value>*
Set for this project

bundle config get *<name>*
Show one setting

bundle config set --local path vendor/bundle
Install gems into the project

bundle config set --local deployment true
Frozen lockfile + vendor/bundle

bundle config set --local frozen true
Fail if the lockfile would change

bundle config set --local without 'development test'
Skip groups on install

GEMFILE SYNTAX

source "https://rubygems.org"
Where gems come from

GEMFILE SYNTAX (cont.)

gem "*<name>***"**
Depend on a gem

gem "*<name>***", "~> 1.2"**
Pessimistic: >= 1.2, < 2.0

gem "*<name>***", "~> 1.2.3"**
Pessimistic: >= 1.2.3, < 1.3

gem "*<name>***", require: false**
Install but do not auto-require

gem "*<name>***", group: :test**
Put one gem in a group

group :development, :test do ... end
Group several gems

ruby "*<version>***"**
Pin the Ruby version

gem "*<name>***", path: "***<dir>***"**
Use a local checkout

gem "*<name>***", git: "***<url>***", branch: "***<b>***"**
Use a git repository

IN RUBY CODE

require "bundler/setup"
Put bundle gems on the load path