Bundler Ruby dependencies

START HERE

`bundle` **`init`**
Create a Gemfile

`bundle` **`update`** *`<gem>`*
Upgrade one gem

`bundle` **`add`** *`<gem>`*
Add a gem and install

`bundle` **`outdated`**
See newer versions

`bundle` **`install`**
Install what the lockfile says

`bundle` **`list`**
All gems in the bundle

`bundle` **`exec`** *`<cmd>`*
Run a command in the bundle

`bundle` **`config set --local`** *`<name> <value>`*
Change a setting

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

`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?

`bundle` **`open`** *`<gem>`*
Open a gem's source in $EDITOR

UPDATE & OUTDATED

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

`bundle` **`update --all`**
Upgrade everything

With the update_requires_all_flag setting on, a bare bundle update is refused; --all makes the intent explicit.

`bundle` **`outdated`**
List gems with newer versions

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

`bundle` **`update --patch`**
Prefer patch releases only

`bundle` **`update --minor`**
Prefer minor releases at most

`bundle` **`update --bundler`**
Update the locked Bundler version

`bundle` **`outdated --only-explicit`**
Only gems named in the Gemfile

INSPECT

`bundle` **`list`**
Every gem in the bundle with version

`bundle` **`list --name-only`**
Just the names

`bundle` **`list --only-group`** *`<group>`*
Gems of one group

`bundle` **`info`** *`<gem>`*
Details about one gem

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

`bundle` **`show`** *`<gem>`*
Path of a gem

LOCKFILE

`bundle` **`lock`**
Resolve and write the lockfile only

`bundle` **`lock --update`** *`<gem>`*
Re-resolve a single gem

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

For example x86_64-linux or arm64-darwin, so one lockfile works on CI and laptops.

`bundle` **`lock --add-checksums`**
Add a CHECKSUMS section

CONFIG & DEPLOY

`bundle` **`config`**
List all settings and where set

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

`bundle` **`config set --global`** *`<name> <value>`*
Set for your user

`bundle` **`config get`** *`<name>`*
Show one setting

`bundle` **`config unset`** *`<name>`*
Remove a 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

`bundle` **`config set --local with`** *`<group>`*
Install an optional group

`bundle` **`config set --local force_ruby_platform true`**
Compile native gems from source

`bundle` **`cache`**
Copy gems into vendor/cache

GEMFILE SYNTAX

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

`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>`***`", "⩾ 1.0", "< 2.0"`**
Several constraints

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

`gem "`*`<name>`***`", require: "`***`<path>`***`"`**
Require path differs from the name

GEMFILE SYNTAX (cont.)

`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

`gem "`*`<name>`***`", github: "`***`<user>`***`/`***`<repo>`***`"`**
GitHub shorthand

`gem "`*`<name>`***`", platforms: :windows`**
Only on given platforms

`gem "`*`<name>`***`", source: "`***`<url>`***`"`**
Take one gem from another source

`source "`*`<url>`***`" do ... end`**
Gems from a private source

`gemspec`
Take dependencies from the .gemspec

IN RUBY CODE

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

`Bundler.require(:`*`<group>`***`)`**
Require the gems of listed groups

`BUNDLE_GEMFILE=`*`<path>`*** *`<cmd>`*
Use another Gemfile for one command

RECIPES

New project

```
bundle init
bundle add rake
bundle add rspec -g test
bundle exec rspec
```

CI or production install

```
bundle config set --local
deployment true
bundle config set --local
without 'development test'
bundle install
```

Make the lockfile work everywhere

```
bundle lock --add-platform
x86_64-linux
bundle lock --add-platform
arm64-darwin
git add Gemfile.lock
```