buf Protobuf toolkit

START HERE

`buf config init`
Create buf.yaml

`buf breaking --against <input>`
Detect breaking changes

`buf build`
Compile and check protos

`buf generate`
Generate code

`buf lint`
Check style rules

`buf dep update`
Pin dependencies

`buf format -w`
Format files in place

`buf curl <url>`
Call an RPC

proto files buf.yaml module → `buf lint / format` → **Checked source** style ok → `buf breaking` → **Compatible** vs. last release → `buf generate` → **Generated code** buf.gen.yaml

CONCEPTS

buf reads buf.yaml for modules, lint and breaking rules, and buf.gen.yaml for code generation. Inputs default to the current directory.

An input can be a directory, a git ref, a tarball, an image file or a BSR module: the same argument works for build, lint, breaking and generate.

Config is v2 (version: v2). Run buf config migrate to upgrade an older v1 buf.yaml and buf.gen.yaml.

SETUP AND CONFIG

`buf --version`
Print the installed version

`buf config init`
Create a buf.yaml here

`buf config ls-lint-rules`
List available lint rules

`buf config ls-breaking-rules`
List available breaking rules

`buf config ls-modules`
List modules in the workspace

`buf config migrate`
Upgrade config files to v2

`buf <command> --help`
Help for any command

BUILD

`buf build`
Compile all protos in the module
Fails with the same errors protoc would report; a quick sanity check in CI.

`buf build -o image.binpb`
Write a binary image

`buf build --as-file-descriptor-set -o fds.binpb`
Write a plain FileDescriptorSet

`buf build --path <dir/file.proto>`
Build only part of the module

`buf build --type <pkg.Message>`
Keep only one type and its deps

LINT

`buf lint`
Lint the current module

`buf lint <input>`
Lint a dir, git ref or image

LINT (cont.)

`buf lint --path <file.proto>`
Lint selected files only

`buf lint --exclude-path <path>`
Skip files or directories

`buf lint --error-format github-actions`
Annotations for GitHub Actions

`buf lint --error-format json`
Machine-readable violations

`buf lint --error-format config-ignore-yaml`
Print ignore_only YAML for violations
Paste the output into buf.yaml to baseline existing violations.

Lint categories

Category	Meaning
STANDARD	Default, recommended rules
BASIC	Smaller subset of STANDARD
MINIMAL	Package layout only
COMMENTS	Require comments (opt in)
UNARY_RPC	Forbid streaming RPCs (opt in)

`// buf:lint:ignore <RULE_ID>`
Silence one rule above a line
Place it directly before the element; disallow_comment_ignores: true turns it off.

FORMAT

`buf format`
Print formatted files to stdout

`buf format -w`
Rewrite files in place

`buf format -d`
Show a diff instead

`buf format -d --exit-code`
Fail if anything is unformatted (CI)

BREAKING CHANGES

`buf breaking --against '.git#branch=main'`
Compare with main in git

`buf breaking --against <dir>`
Compare with another directory

`buf breaking --against image.binpb`
Compare with a saved image

`buf breaking --against-registry`
Compare with the latest BSR commit

BREAKING CHANGES (cont.)

`buf breaking --against <input> --path <dir>`
Check only part of the module

`buf breaking --against <input> --error-format github-actions`
Annotations for CI

Breaking categories

Category	Catches changes that break
FILE	Generated source per file (default)
PACKAGE	Generated source per package
WIRE_JSON	Wire and JSON encoding
WIRE	Wire encoding only

`buf build -o baseline.binpb`
Save a baseline image of a release

GENERATE

`buf generate`
Run plugins from buf.gen.yaml

`buf generate --template <file.yaml>`
Use another template file

`buf generate --path <dir>`
Generate for part of the module

`buf generate --include-imports`
Also generate code for imports
Well-known types are not included.

`buf generate --clean`
Delete plugin output dirs first

`buf generate <input>`
Generate from a git repo or module

DEPENDENCIES AND MODULES

`buf dep update`
Resolve deps and write buf.lock

`buf export <input> -o <dir>`
Copy protos and deps into a dir

CONVERT AND CALL

`buf convert <input> --type <pkg.Msg> --from <file>.json --to <file>.binpb`
Convert a message between formats

`buf curl --list-services <url>`
List services via reflection

`buf curl --list-methods <url>`
List methods via reflection

CONVERT AND CALL (cont.)

`buf curl -d '<json>' <url>/<pkg.Svc>/<Method>`
Call an RPC with a JSON body

`buf curl -d @<file>.json <url>/<pkg.Svc>/<Method>`
Read the body from a file

`buf curl --schema <dir> <url>/<pkg.Svc>/<Method>`
Use local protos, no reflection

`buf curl --protocol grpc <url>/<pkg.Svc>/<Method>`
Speak gRPC (default is Connect)

`buf curl -H "Authorization: Bearer <token>" <url>/<pkg.Svc>/<Method>`
Send a header

`buf curl --http2-prior-knowledge <url>/<pkg.Svc>/<Method>`
Plaintext HTTP/2 (h2c)

`buf curl -v <url>/<pkg.Svc>/<Method>`
Show request and response details

CONFIG FILES

buf.yaml

```
version: v2
modules:
  - path: proto
deps:
  - buf.build/googleapis/googleapis
lint:
  use:
    - STANDARD
  except:
    - PACKAGE_VERSION_SUFFIX
breaking:
  use:
    - FILE
```

buf.gen.yaml

```
version: v2
clean: true
plugins:
  - remote: buf.build/protocolbuffers/go
    out: gen/go
    opt: paths=source_relative
  - local: protoc-gen-go-grpc
    out: gen/go
inputs:
  - directory: proto
```

RECIPES

CI checks

```
buf format -d --exit-code
buf lint
buf build
buf breaking --against '.git#branch=main'
```

Fresh code generation

```
buf dep update
buf generate --clean
git diff --exit-code gen/
```