Nushell Structured data shell

START HERE

`ls | where size > 1mb`
Filter rows

`get <col>`
Extract a value

`open <file>`
Load as data

`let <name> = <value>`
Set a variable

`select <col> ...`
Pick columns

`def <name> [args] { }`
Custom command

`sort-by <col>`
Sort rows

`help <command>`
Built-in docs

Source ls, open, ps → **Table** rows + columns `where` / `select` → **Filtered** still structured `to json` / `save` → **Output** text or file

CONCEPTS

Pipelines pass typed data (tables, records, lists), not text, so filter by column instead of parsing with grep and awk.

Types: int, float, string, bool, filesize (1mb), duration (2hr), date, list, record, table, closure, nothing.

Nushell is not POSIX: no $VAR, no &&, no backtick substitution. External commands still run and return text.

NAVIGATE & FILES

`ls`
List files as a table

`ls -a`
Include hidden files

`ls **/*.rs`
Glob recursively

`cd <dir>`
Change directory

`mkdir <dir>`
Create directory with parents

`cp -r <src> <dst>`
Copy recursively

`mv <src> <dst>`
Move or rename

`rm -r <path>`
Remove recursively
rm sends to trash by default on most platforms; use -p to delete permanently

OPEN & CONVERT

`open <file>`
Load file, parsed by extension

`open --raw <file>`
Read as plain text

`save <file>`
Write pipeline to file

`from json / from yaml / from csv`
Parse text into data

`to json / to yaml / to csv`
Render data as text

OPEN & CONVERT (cont.)

`http get <url>`
Fetch URL, parse JSON

FILTER & SHAPE

`where <col> == <val>`
Keep matching rows

`where {|r| $r.size > 1kb }`
Filter with a closure

`select <col> <col>`
Keep columns

`reject <col>`
Drop columns

`get <col>`
Pull out column or field

`get a.b.0`
Cell path into nested data

`sort-by <col>`
Sort rows

`sort-by -r <col>`
Sort descending

`first <n>` / `last <n>`
Take rows from either end

`skip <n>`
Drop first rows

`each {|x| ... }`
Map over items

`length`
Count items

`update <col> {|r| ... }`
Change a column

`insert <col> {|r| ... }`
Add a new column

STRINGS & TEXT

`$"hello ($name)"`
Interpolate expressions

`str replace <a> <b>`
Replace text

`str replace -r <re> <b>`
Replace by regex

`str contains <text>`
Substring test

`split row <sep>`
String to list

`lines`
Split text into lines

`=~ / !~`
Regex match operators

VARIABLES & FLOW

`let <name> = <value>`
Immutable variable

`mut <name> = <value>`
Mutable variable

`if <cond> { } else { }`
Conditional (an expression)

`for i in 1..5 { print $i }`
Loop over range or list

`try { } catch { }`
Handle errors

`$in`
Pipeline input inside a block

`{a: 1, b: 2}`
Record literal

`[1 2 3]`
List literal, space separated

COMMANDS & MODULES

`def <name> [x: int] { }`
Custom command

`def <name> [--flag, --opt: string] { }`
Flags and options

`def <name> [x?: int] { }`
Optional parameter

`{|x| $x * 2 }`
Closure

`alias <name> = <cmd>`
Short alias

`use <module>`
Load a module

`nu <script.nu>`
Run a script file

`help commands`
Search all commands

`help <command>`
Help for one command

ENVIRONMENT & CONFIG

`$env.FOO = "bar"`
Set environment variable

`$env.FOO?`
Read variable, null if unset

`$env.PATH ++= ["~/bin"]`
Append to PATH

`config nu`
Edit config.nu

EXTERNALS & I/O

`^<cmd>`
Force the external command
Use ^ssh or ^ls to bypass a same-named built-in

`cmd | complete`
Capture stdout, stderr, exit code

BASH TO NUSHELL

Common translations

Bash	Nushell
`export FOO=bar`	`$env.FOO = "bar"`
`echo $FOO`	`$env.FOO`
`a && b`	`a; b`
`a || b`	`try { a } catch { b }`
`echo "x $(cmd)"`	`$"x (cmd)"`
`cmd | head -5`	`cmd | first 5`
`cat file`	`open --raw file`
`unset FOO`	`hide-env FOO`
`find . -name '*.rs'`	`ls **/*.rs`
`cmd > file 2>&1`	`cmd o+e> file`

Multi-line commands need no backslash: wrap them in parentheses or end a line with an open pipe.

RECIPES

Biggest files

```
ls **/* | where type == file | sort-by size | reverse | first 10 | select name size
```

Edit JSON in place

```
open package.json | update version "2.0.0" | save -f package.json
```

API to CSV

```
http get https://api.github.com/repos/nushell/nushell/releases
  | select tag_name published_at
  | first 5 | to csv
```

Custom command with flag

```
def greet [name: string, --loud] {
  if $loud { $name | str upcase } else { $name }
}
```