

## Grouping and Sections

Well-designed CLIs group related output under clear headings or sections. For example, the Heroku CLI uses section headers (“=== ...”) to label blocks:

```
$ heroku apps
=== user@example.org Apps
example
example2

=== Collaborated Apps
collabapp                owner@example.org
```

Here each “===” header labels a section (owned vs. collaborated apps) <sup>1</sup>. Within a section, CLI output is often shown as aligned key/value pairs or lists. For instance, `heroku apps:info` prints the app name as a heading, then indented fields under it:

```
$ heroku apps:info
=== example-app-69977
Auto Cert Mgmt: false
Dynos:
Git URL:         https://git.heroku.com/example-app-69977.git
Owner:           you@example.com
Region:          us
...
```

This groups each field under the app name with consistent padding on the left of each value <sup>2</sup>. In general, related items are clustered together (e.g. status lines, metadata, configuration) with logical subheaders or indentation. As the CLI Guidelines advise, “When a command changes the state, it’s especially valuable to explain what has just happened” so state changes are grouped in their own output block <sup>3</sup>.

## Whitespace and Alignment

High-quality CLIs use whitespace liberally to align columns and separate blocks. In tabular outputs, column headings line up over data columns, and separators are used. For example, Azure CLI’s table output aligns headers over columns with dash lines:

Name	ResourceGroup	Location
-----	-----	-----

DemoVM010	DEMORG1	westus
KBDemo001VM	RGDEMO001	westus

The dashed line under headers (equal length to the headers) makes the columns easy to scan <sup>4</sup>. Likewise, Docker's output (`docker ps`) uses fixed-width columns separated by spaces:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			
919e1179bdb8	ubuntu:c1	"top"	1 minute ago	Up 1
minute ...	admiring_lovelace			

Each column is padded so that items align vertically <sup>5</sup>. Even when text wraps or is long, best practice is to keep one record per line (or use an alternate "plain" mode) so that tools like `grep` can still work <sup>6</sup>.

Vertical spacing (blank lines) is used to separate logical chunks. In [28], there's a blank line between the "Apps" section and the "Collaborated Apps" section to visually separate them. In summary outputs (like help messages or lists of subcommands), commands or sections are usually separated by a blank line for readability. In error or log outputs, blank lines can separate an error summary from details or hints.

## Output Ordering (Summary vs Detail)

Guidelines recommend putting high-level status or summary information where it's most visible. Many tools print progress or summary first, then details. For example, `git push` (cited in the CLI Guidelines) prints many progress lines *first* (enumerating, counting, compressing), ending with a final summary of what was pushed <sup>7</sup>. This shows intermediate steps followed by a conclusion. Similarly, `git status` first shows branch info (summary), then sections of changes, and finally suggestions at the end <sup>8</sup>.

Conversely, the clig.dev guide notes that *final outcomes* or errors should stand out: "Put the most important information at the end of the output" because the eye is drawn to the bottom (and to colored text) <sup>9</sup>. In practice, this means an action's final "done" message, or an error message, comes after any diagnostic details. Errors and warnings are typically printed last (and to stderr); for example, Heroku's action commands print progress on stderr with a "done" at the end <sup>10</sup>. Color (red for errors) and placement at the end make problems easy to notice <sup>9</sup> <sup>10</sup>.

## Layout Conventions

- **Tabular output.** Many tools present lists of items as tables. They include a header row, underlined or separated from data. Azure CLI's table format (above) and AWS CLI's `--output table` use ASCII tables with borders (using `+`, `-`, `|`) <sup>11</sup>. Kubernetes' `kubectl get pods` similarly prints a table (no borders) with columns like NAME, READY, STATUS, etc. The key is consistent column widths. Numeric or date columns are often right-aligned, text left-aligned.

- **Lists and menus.** When presenting a choice or list, tools use bullets or prompts. For example, Heroku’s interactive SSH-key prompt shows a bullet list with an arrow cursor:

```
$ heroku keys:add
heroku keys:add
? Which SSH key would you like to upload? (Use arrow keys)
> /Users/alice/.ssh/id_rsa.pub
  /Users/alice/.ssh/id_work.pub
  /Users/alice/.ssh/id_other.pub
```

Here each option is a line, and the selected line is prefixed with an arrow (➤) <sup>12</sup>. This menu-style layout cleanly groups choices with no excess text.

- **Summaries.** When applicable, commands often begin or end with a brief summary. For example, after an action the CLI might print “Done” or “XYZ complete” (Heroku’s `maintenance:on` prints “Enabling maintenance mode... done” <sup>13</sup>). Status commands (e.g. `kubectl get service`) typically start with a header line summarizing the object being listed or the current context.
- **Logs.** When outputting logs or streaming data, CLIs often timestamp or clearly label each line. While not the focus here, logs typically align timestamps or levels (INFO, ERROR) in fixed columns. The same alignment/spacing principles apply: same width for dates, indent for details, clear separators if multilines.
- **Section dividers.** Horizontal rules or divider lines can separate sections. In Heroku’s regions example, after a header line the columns are underlined with “—” characters <sup>14</sup>. Azure and AWS CLI table modes use dashed lines with `-` or `+` to separate headers from data <sup>4</sup> <sup>11</sup>. These visual rules help the eye track columns. Some CLIs (e.g. Heroku’s help output) even use repeated “====” as a divider or section title.
- **Bullet points.** Bullets are more common in documentation or help text than in normal output, but some CLIs print lists of items with simple markers. When listing multiple suggestions or items, a dash or asterisk can be used (e.g. an error message might list steps with “\*” or “-”). The key is to align bullets and indent wrapped lines so the text lines up under the bullet.

## Official Guidelines and Examples

CLI style guides reinforce these practices. The Heroku CLI guide emphasizes “usability first” with human-readable output, urging consistent formatting and the option of machine-readable modes (JSON or terse flags) <sup>15</sup>. It explicitly shows good table layout: e.g. switching from a human-unfriendly dual-section list to a single table with columns (“ID”, “Location”, “Runtime”) under a line of dashes <sup>14</sup>. Heroku’s examples use aligned columns and underline separators so output remains `grep`-friendly <sup>14</sup>.

The open-source CLI guidelines (clig.dev) echo these points: humans before machines, but offer JSON/`--plain` modes for scripts <sup>16</sup> <sup>17</sup>. They note that multiline or richly formatted tables should have a plain

fallback to preserve one-record-per-line <sup>18</sup>. The guidelines also recommend using color sparingly (e.g. red for errors) and disabling it when not appropriate <sup>19</sup>, and to keep success output brief—often just a final status line <sup>20</sup>.

Microsoft's Azure CLI docs show that even default output is JSON (for scriptability), but the `--output table` mode produces a neat ASCII table for humans <sup>21</sup>. For example, `az vm list --output table` prints:

Name	ResourceGroup	Location
-----	-----	-----
DemoVM010	DEMORG1	westus
KBDemo001VM	RGDEMO001	westus

which is easy to scan <sup>4</sup>. AWS CLI similarly supports a `table` format that uses `+`/`-` borders for readability. In an example `aws ec2 describe-volumes --output table`, the AWS docs show:

-----				
	DescribeVolumes			
+-----+	+-----+			+
AZ	ID	InstanceId	Size	
+-----+	+-----+			+
us-west-2a	vol-e11a5288	i-a071c394	30	
us-west-2a	vol-2e410a47	i-4b41a37c	8	
+-----+	+-----+			+

This fully-bordered table (headers, columns, separators) makes the output self-contained and human-friendly <sup>11</sup>.

**Best Practices Summary (from style guides):** Commands should clearly separate sections, align columns, and use whitespace and lines to improve scanning. Include headings or dividers for context. Order output logically (steps then conclusion, or summary then details) with errors at the end and highlighted. Offer machine-readable flags (`--json`, `--quiet`) but default to concise, user-friendly display. Use color only for emphasis. Many guides stress consistency and simplicity (e.g. GNU and Heroku recommend lower-case, no punctuation, and fitting within ~80 columns for legibility). Examples above demonstrate these principles in action <sup>3</sup> <sup>14</sup> <sup>4</sup>, producing CLI output that is grouped, aligned, and easy to scan.

**References:** Official CLI style guides and documentation (Heroku Dev Center, Azure CLI docs, AWS CLI User Guide, clig.dev) provide detailed recommendations and example outputs <sup>3</sup> <sup>14</sup> <sup>4</sup> <sup>11</sup> <sup>9</sup>. These sources illustrate well-formatted CLI output and the reasoning behind layout choices.

---

<sup>1</sup> <sup>2</sup> CLI Usage | Heroku Dev Center  
<https://devcenter.heroku.com/articles/using-the-cli>

3 6 7 8 9 16 17 18 19 20 **Command Line Interface Guidelines**

<https://clig.dev/>

4 21 **Output formats for Azure CLI commands | Microsoft Learn**

<https://learn.microsoft.com/en-us/cli/azure/format-output-azure-cli?view=azure-cli-latest>

5 **docker container ls | Docker Docs**

<https://docs.docker.com/reference/cli/docker/container/ls/>

10 12 13 14 15 **CLI Style Guide | Heroku Dev Center**

<https://devcenter.heroku.com/articles/cli-style-guide>

11 **Setting the output format in the AWS CLI - AWS Command Line Interface**

<https://docs.aws.amazon.com/cli/v1/userguide/cli-usage-output-format.html>