# building amazing CLI with oclif _

Gaurav Chodwadia

Software Engineer at PayPal

# We'll talk about_

- **Why** to build CLI_
- **How** to design one_
- **What** is oclif _
- **How** to create an extensible CLI_
- **How** to beautify_
- **What? How? Why? When? Where?_**

# Why to build CLI_

- **CLI Superpowers_**
  - For the users
    - Quick to use
    - Repeatable and Composable Tasks
  - For the developers
    - cURL vs CLI
    - Quick to build compared to Web UI


Snap-on
SEARCHING
https://nmap.org/6/
Image courtesy of Fuel VFX
copyright: © 2013 Marvel Studios

# How to design one_

- Command Structure_

```
~
$ aws
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

  aws help
  aws <command> help
  aws <command> <subcommand> help
aws: error: too few arguments
```

# How to design one_

- Command Structure_

```
~
$ gcloud
ERROR: (gcloud) Command name argument expected.
```

```
Command name argument expected.

Available commands for gcloud:

  AI and Machine Learning
      ai-platform               Manage AI Platform jobs and models.
      ml                        Use Google Cloud machine learning capabilities.
      ml-engine                 Manage AI Platform jobs and models.

  API Platform and Ecosystems
      endpoints                 Create, enable and manage API services.
      service-management        Create, enable and manage API services.
```

# How to design one_

- Command Structure_

```
~
$ firebase
Usage: firebase [options] [command]

Options:
  -V, --version                                output the version number
  -P, --project <alias_or_project_id>          the Firebase project to use for this command
  -j, --json                                   output JSON instead of text, also triggers non-interactive mode
  --token <token>                              supply an auth token for this command
  --non-interactive                            error out of the command instead of waiting for prompts
  --interactive                                force interactive shell treatment even when not detected
  --debug                                      print verbose debug output and keep a debug log file
  -h, --help                                   output usage information

Commands:
  auth:export [options] [dataFile]             Export accounts from your Firebase project into a data file
  auth:import [options] [dataFile]             import users into your Firebase project from a data file(.csv or .json)
  database:get [options] <path>                fetch and print JSON data at the specified path
  database:instances:create <instanceName>     create a realtime database instance
  database:instances:list                      list realtime database instances
  database:profile [options]                   profile the Realtime Database and generate a usage report
  database:push [options] <path> [infile]      add a new JSON object to a list of data in your Firebase
  database:remove [options] <path>             remove data from your Firebase at the specified path
  database:set [options] <path> [infile]       store JSON data at the specified path via STDIN, arg, or file
  database:settings:get [options] <path>       read the realtime database setting at path
```

# How to design one_

- Command Structure_

```
~
$ heroku
CLI to interact with Heroku

VERSION
  heroku/7.29.0 darwin-x64 node-v11.14.0

USAGE
  $ heroku [COMMAND]

COMMANDS
  access          manage user access to apps
  addons          tools and services for developing, extending, and operating your app
  apps            manage apps on Heroku
  auth            check 2fa status
  authorizations  OAuth authorizations
  autocomplete    display autocomplete installation instructions
  base
  buildpacks      scripts used to compile apps
  certs           a topic for the ssl plugin
  ci              run an application test suite on Heroku
```

# How to design one_

- Command Structure_

```
~
$ heroku access --help
list who has access to an app

USAGE
  $ heroku access

OPTIONS
  -a, --app=app          (required) app to run command against
  -r, --remote=remote    git remote of app to use
  --json                 output in json format

COMMANDS
  access:add     add new users to your app
  access:remove  remove users from a team app
  access:update  update existing collaborators on an team app
```

# How to design one_

- Command Structure_
- Args vs Flags_

380

The syntax for `scp` is:

If you are on the computer from which you want to send file to a remote computer:

```
scp /file/to/send username@remote:/where/to/put
```

Here the `remote` can be a FQDN or an IP address.

On the other hand if you are on the computer wanting to receive file from a remote computer:

```
scp username@remote:/file/to/send /where/to/put
```

`scp` can also send files between two remote hosts:

```
scp username@remote_1:/file/to/send username@remote_2:/where/to/put
```

So the basic syntax is:

```
scp username@source:/location/to/file username@destination:/where/to/put
```

You can read `man scp` to get more ideas on this.

# **How** to design one_

- Command Structure_
- Args vs Flags_

> Funny story about -v. I was using pkill to get rid of some hung processes on a remote server. For some reason I decided to get verbose output and run pkill -v.
>
> Turns out -v negates the match so I killed every process on the box but one.
>
> I had to email someone to reboot the server.

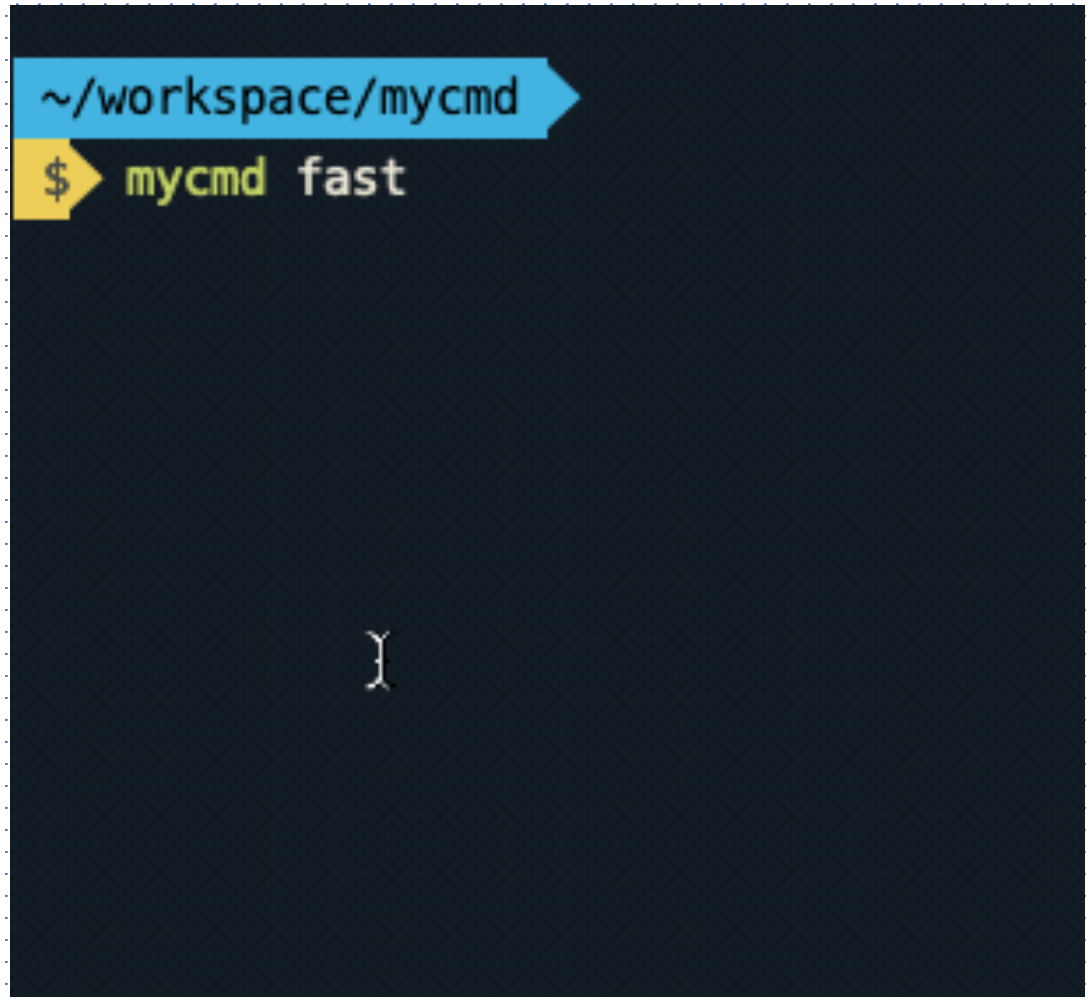
# How to design one_

- Command Structure_
- Args vs Flags_
- Use Prompts_

- https://www.npmjs.com/package/inquirer
- https://www.npmjs.com/package/enquirer

# How to design one_

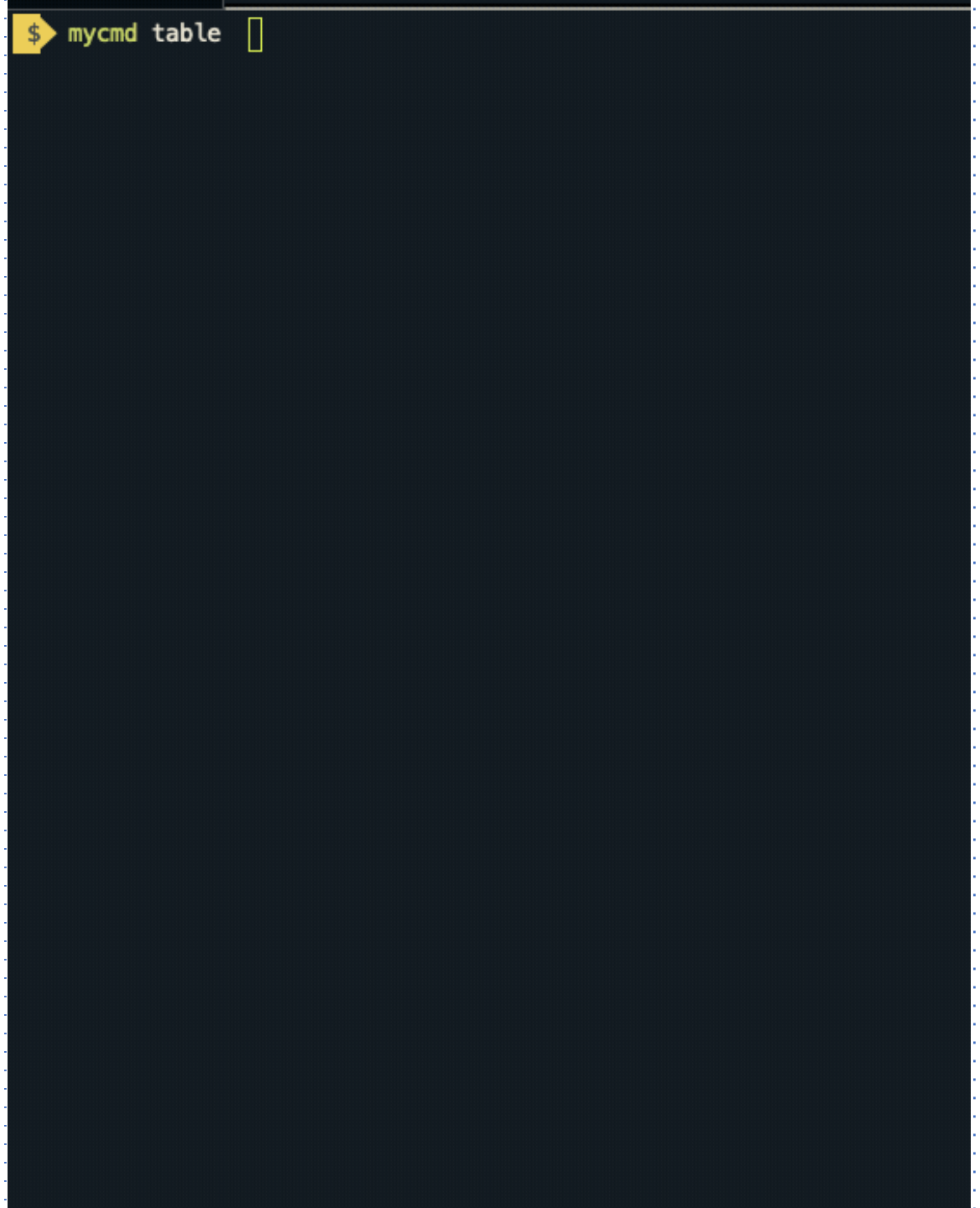
- Command Structure_
- Args vs Flags_
- Use Prompts
- Spinners




https://www.npmjs.com/package/cli-spinners

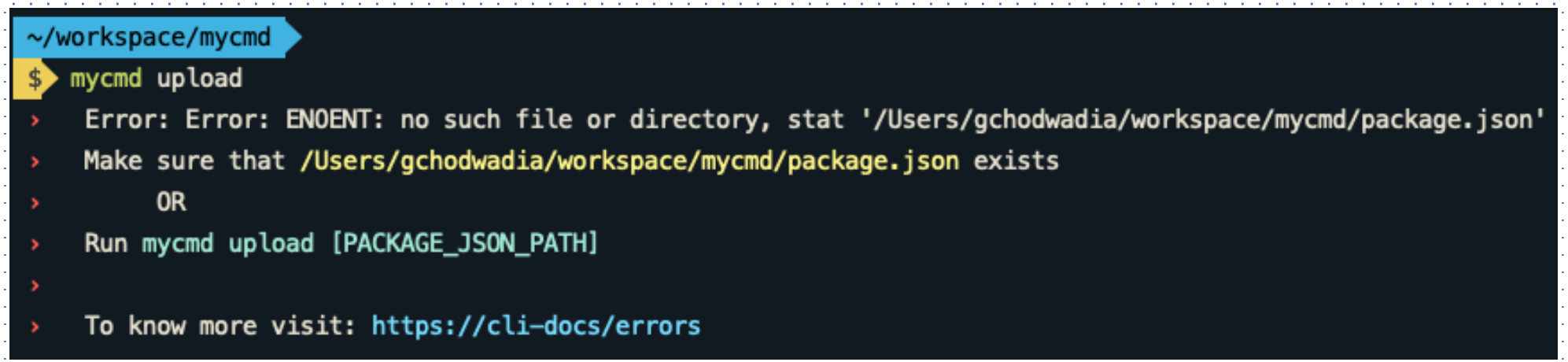
# How to design one_

- Command Structure_
- Args vs Flags_
- Use Prompts
- Spinners
- Tables



# How to design one_

- Command Structure_
- Args vs Flags_
- Use Prompts
- Spinners
- Tables
- Errors



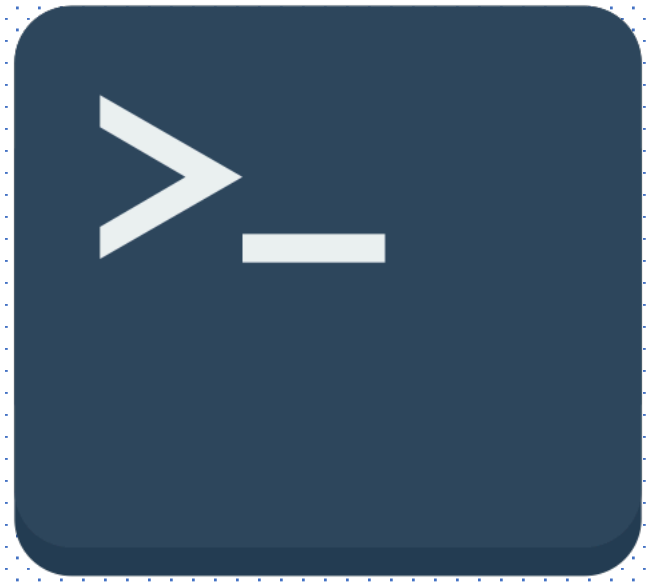
# What is oclif _

## Open cli framework

Made with 💜 by Heroku — MIT License

- Scaffolding
- Single / Multi Commands
- Plugins
- Hooks
- Parsing Args and Flags
- Auto Documentation
- Configurations
- Releasing
- Testing
- TypeScript

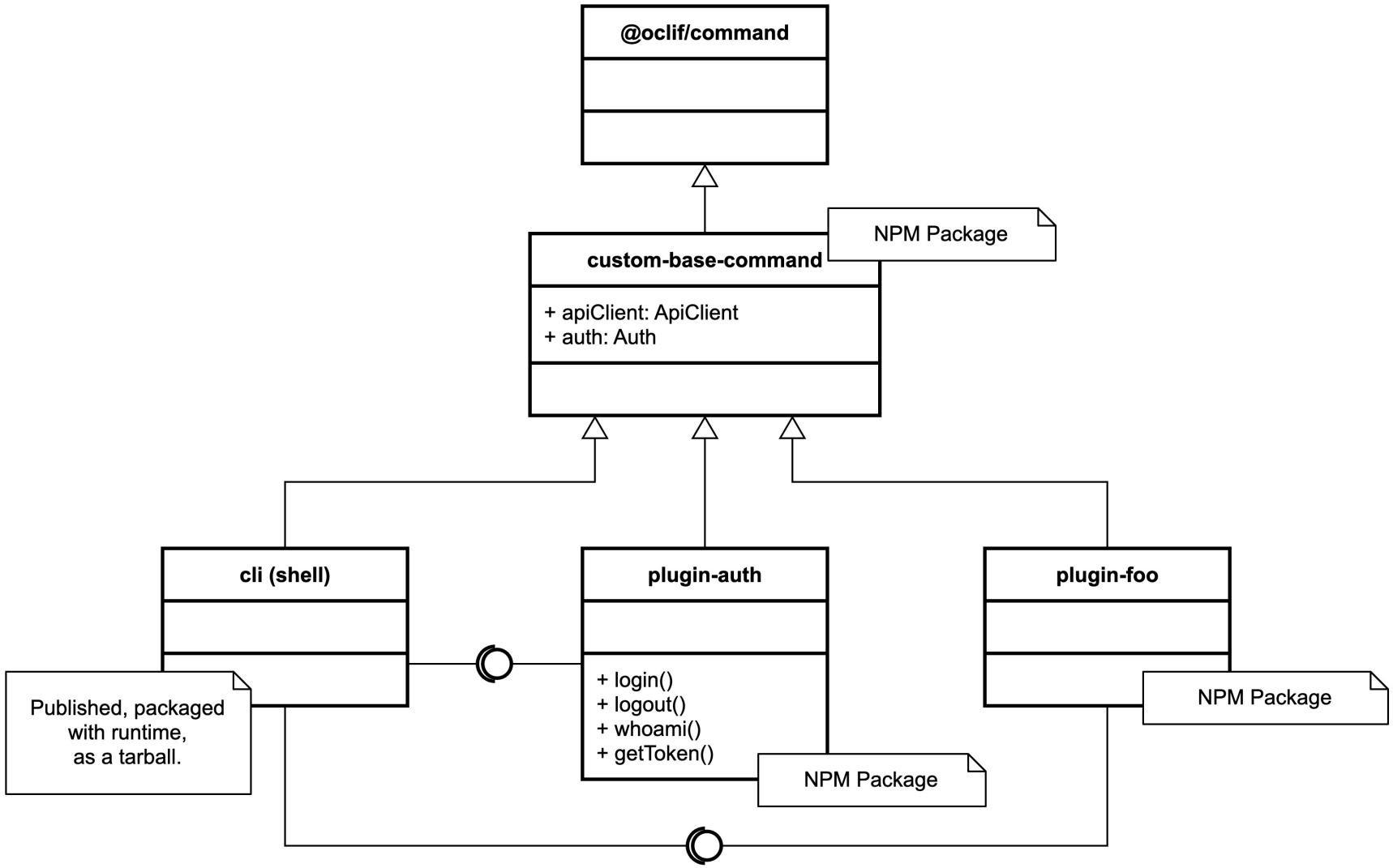
# How to create an extensible CLI_



# **What** we're doing at PayPal_

- Developer Console
- Federated Development
- Everything extension
- Extensible CLI

# What we did at PayPal_



# What else_

@oclif/plugin-not-found - Display a friendly "did you mean" message if a command is not found.
@oclif/plugin-plugins - Allow users to add plugins to extend your CLI.
@oclif/plugin-update - Add autoupdate support to the CLI.
@oclif/plugin-help - Help plugin for oclif.
@oclif/plugin-warn-if-update-available - Show a warning message if user is running an out of date CLI.
@oclif/plugin-which - Show which plugin a command comes from.
@oclif/plugin-commands - Add a commands command to list all the commands.
@oclif/plugin-autocomplete - Add bash/zsh autocomplete.

@oclif/command - Base command for oclif. This can be used directly without the generator.
@oclif/config - Most of the core setup for oclif lives here.
@oclif/errors - Renders and logs errors from commands.
@oclif/cli-ux - Library for common CLI UI utilities.
@oclif/test - Test helper for oclif.

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

# What? How? Why? When? Where?_

# Thank You

https://www.linkedin.com/in/gauravchodwadia/

https://twitter.com/Gaurav51289