Despite the lack of feature parity at the moment, Arduino CLI provides many of the features you can find in the Arduino IDE. Let's see some examples.

## Before you start

`arduino-cli` is a container of commands and each command has its own dedicated help text that can be shown with the `help` command like this:

```
$ arduino-cli help core
Arduino Core operations.

Usage:
    arduino-cli core [command]

Examples:
    ./arduino-cli core update-index

Available Commands:
    download     Downloads one or more cores and corresponding tool dependencies.
    install      Installs one or more cores and corresponding tool dependencies.
    list         Shows the list of installed platforms.
    search       Search for a core in Boards Manager.
    uninstall    Uninstalls one or more cores and corresponding tool dependencies if no more used.
    update-index Updates the index of cores.
    upgrade      Upgrades one or all installed platforms to the latest version.

Flags:
    -h, --help   help for core

Global Flags:
        --additional-urls strings   Additional URLs for Boards Manager.
        --config-file string        The custom config file (if not specified the default will be used).
        --format string             The output format, can be [text|json]. (default "text")
        --log-file string           Path to the file where logs will be written.
        --log-format string         The output format for the logs, can be [text|json].
        --log-level string          Messages with this level and above will be logged.
    -v, --verbose                   Print the logs on the standard output.

Use "arduino-cli core [command] --help" for more information about a command.
```

## Create a configuration file

Arduino CLI doesn't strictly require a configuration file to work because the command line interface provides any possible functionality. However, having one can spare you a lot of typing when issuing a command, so let's go ahead and create it with:

```
$ arduino-cli config init
Config file written: /home/luca/.arduino15/arduino-cli.yaml
```

If you inspect the contents of `arduino-cli.yaml`, you'll find the available options with their respective default values. For more information, see the configuration documentation.

## Create a new sketch

To create a new sketch named `MyFirstSketch` in the current directory, run the following command:

```
$ arduino-cli sketch new MyFirstSketch
Sketch created in: /home/luca/MyFirstSketch
```

A sketch is a folder containing assets like source files and libraries; the `new` command creates for you a .ino file called `MyFirstSketch.ino` containing Arduino boilerplate code:

```
$ cat $HOME/MyFirstSketch/MyFirstSketch.ino
void setup() {
}

void loop() {
}
```

At this point you can use your favourite file editor or IDE to open the file `$HOME/MyFirstSketch/MyFirstSketch.ino` and change the code like this:

```
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
}
```

## Connect the board to your PC

The first thing to do upon a fresh install is to update the local cache of available platforms and libraries by running:

```
$ arduino-cli core update-index
Updating index: package_index.json downloaded
```

After connecting the board to your PC by using the USB cable, you should be able to check whether it's been recognized by running:

```
$ arduino-cli board list
Port         Type              Board Name              FQBN                Core
/dev/ttyACM1 Serial Port (USB) Arduino/Genuino MKR1000 arduino:samd:mkr1000 arduino:samd
```

In this example, the MKR1000 board was recognized and from the output of the command you see the platform core called `arduino:samd` is the one that needs to be installed to make it work.

If you see an `Unknown` board listed, uploading should still work as long as you identify the platform core and use the correct FQBN string. When a board is not detected for whatever reason, you can list all the supported boards and their FQBN strings by running the following:

```
$ arduino-cli board listall mkr
Board Name              FQBN
Arduino MKR FOX 1200    arduino:samd:mkrfox1200
Arduino MKR GSM 1400    arduino:samd:mkrgsm1400
Arduino MKR WAN 1300    arduino:samd:mkrwan1300
Arduino MKR WiFi 1010   arduino:samd:mkrwifi1010
Arduino MKRZERO         arduino:samd:mkrzero
Arduino/Genuino MKR1000 arduino:samd:mkr1000
```

## Install the core for your board

To install the `arduino:samd` platform core, run the following:

```
 $ arduino-cli core install arduino:samd
Downloading tools...
arduino:arm-none-eabi-gcc@4.8.3-2014q1 downloaded
arduino:bossac@1.7.0 downloaded
arduino:openocd@0.9.0-arduino6-static downloaded
arduino:CMSIS@4.5.0 downloaded
arduino:CMSIS-Atmel@1.1.0 downloaded
arduino:arduinoOTA@1.2.0 downloaded
Downloading cores...
arduino:samd@1.6.19 downloaded
Installing tools...
Installing platforms...
Results:
arduino:samd@1.6.19 - Installed
arduino:arm-none-eabi-gcc@4.8.3-2014q1 - Installed
arduino:bossac@1.7.0 - Installed
arduino:openocd@0.9.0-arduino6-static - Installed
arduino:CMSIS@4.5.0 - Installed
arduino:CMSIS-Atmel@1.1.0 - Installed
arduino:arduinoOTA@1.2.0 - Installed
```

Now verify we have installed the core properly by running:

```
 $ arduino-cli core list
ID              Installed       Latest  Name
arduino:samd    1.6.19          1.6.19  Arduino SAMD Boards (32-bits ARM Cortex-M0+)
```

Great! Now we are ready to compile and upload the sketch.

## Adding 3rd party cores

If your board requires 3rd party core packages to work, you can list the URLs to additional package indexes in the Arduino CLI configuration file.

For example, to add the ESP8266 core, edit the configuration file and change the `board_manager` settings as follows:

```
board_manager:
  additional_urls:
    - https://arduino.esp8266.com/stable/package_esp8266com_index.json
```

From now on, commands supporting custom cores will automatically use the additional URL from the configuration file:

```
 $ arduino-cli core update-index
Updating index: package_index.json downloaded
Updating index: package_esp8266com_index.json downloaded
Updating index: package_index.json downloaded

$ arduino-cli core search esp8266
ID              Version Name
esp8266:esp8266 2.5.2   esp8266
```

Alternatively, you can pass a link to the additional package index file with the `--additional-urls` option, that has to be specified every time and for every command that operates on a 3rd party platform core, for example:

```
 $ arduino-cli  core update-index --additional-urls https://arduino.esp8266.com/stable/package_esp8266com_index.json
Updating index: package_esp8266com_index.json downloaded

$ arduino-cli core search esp8266 --additional-urls https://arduino.esp8266.com/stable/package_esp8266com_index.json
ID              Version Name
esp8266:esp8266 2.5.2   esp8266
```

## Compile and upload the sketch

To compile the sketch you run the `compile` command, passing the proper FQBN string:

```
$ arduino-cli compile --fqbn arduino:samd:mkr1000 MyFirstSketch
Sketch uses 9600 bytes (3%) of program storage space. Maximum is 262144 bytes.
```

To upload the sketch to your board, run the following command, using the serial port your board is connected to:

```
$ arduino-cli upload -p /dev/ttyACM0 --fqbn arduino:samd:mkr1000 MyFirstSketch
No new serial port detected.
Atmel SMART device 0x10010005 found
Device       : ATSAMD21G18A
Chip ID      : 10010005
Version      : v2.0 [Arduino:XYZ] Dec 20 2016 15:36:43
Address      : 8192
Pages        : 3968
Page Size    : 64 bytes
Total Size   : 248KB
Planes       : 1
Lock Regions : 16
Locked       : none
Security     : false
Boot Flash   : true
BOD          : true
BOR          : true
Arduino      : FAST_CHIP_ERASE
Arduino      : FAST_MULTI_PAGE_WRITE
Arduino      : CAN_CHECKSUM_MEMORY_BUFFER
Erase flash
done in 0.784 seconds

Write 9856 bytes to flash (154 pages)
[==============================] 100% (154/154 pages)
done in 0.069 seconds

Verify 9856 bytes of flash with checksum.
Verify successful
done in 0.009 seconds
CPU reset.
```

## Add libraries

If you need to add more functionalities to your sketch, chances are some of the libraries available in the Arduino ecosystem already provide what you need. For example, if you need a debouncing strategy to better handle button inputs, you can try searching for the `debouncer` keyword:

```
$ arduino-cli lib search debouncer
Name: "Debouncer"
    Author: hideakitai
    Maintainer: hideakitai
    Sentence: Debounce library for Arduino
    Paragraph: Debounce library for Arduino
    Website: https://github.com/hideakitai
    Category: Timing
    Architecture: *
    Types: Contributed
    Versions: [0.1.0]
Name: "FTDebouncer"
    Author: Ubi de Feo
    Maintainer: Ubi de Feo, Sebastian Hunkeler
    Sentence: An efficient, low footprint, fast pin debouncing library for Arduino
    Paragraph: This pin state supervisor manages debouncing of buttons and handles transitions between LOW and HIGH state, calling a
    Website: https://github.com/ubidefeo/FTDebouncer
    Category: Uncategorized
    Architecture: *
    Types: Contributed
    Versions: [1.3.0]
Name: "SoftTimer"
    Author: Balazs Kelemen <prampec+arduino@gmail.com>
    Maintainer: Balazs Kelemen <prampec+arduino@gmail.com>
    Sentence: SoftTimer is a lightweight pseudo multitasking solution for Arduino.
    Paragraph: SoftTimer enables higher level Arduino programing, yet easy to use, and lightweight. You are often faced with the pro
    Website: https://github.com/prampec/arduino-softtimer
    Category: Timing
    Architecture: *
    Types: Contributed
    Versions: [3.0.0, 3.1.0, 3.1.1, 3.1.2, 3.1.3, 3.1.5, 3.2.0]
```

Our favourite is `FTDebouncer`, let's install it by running:

```
$ arduino-cli lib install FTDebouncer
FTDebouncer depends on FTDebouncer@1.3.0
Downloading FTDebouncer@1.3.0...
FTDebouncer@1.3.0 downloaded
Installing FTDebouncer@1.3.0...
Installed FTDebouncer@1.3.0
```

## Using the `daemon` mode and the gRPC interface

Arduino CLI can be launched as a gRPC server via the `daemon` command.

The client_example folder contains a sample client code that shows how to interact with the gRPC server. Available services and messages are detailed in the gRPC reference pages.

To provide observability for the gRPC server activities besides logs, the `daemon` mode activates and exposes by default a Prometheus endpoint (http://localhost:9090/metrics) that can be fetched for telemetry data like:

```
# TYPE daemon_compile counter
daemon_compile{buildProperties="",exportFile="",fqbn="arduino:samd:mkr1000",installationID="ed6f1f22-1fbe-4b1f-84be-84d035b6369c",jo

# TYPE daemon_board_list counter
daemon_board_list{installationID="ed6f1f22-1fbe-4b1f-84be-84d035b6369c",success="true"} 1 1580385724833
```

The telemetry settings are exposed via the `telemetry` section in the CLI configuration:

```
telemetry:
  enabled: true
  addr: :9090
```