MAKE IT SO

TD-talk by Mads Johansen

21. February 2019

WHAT IS MAKE?

A build tool that performs tasks according to rules in a Makefile.

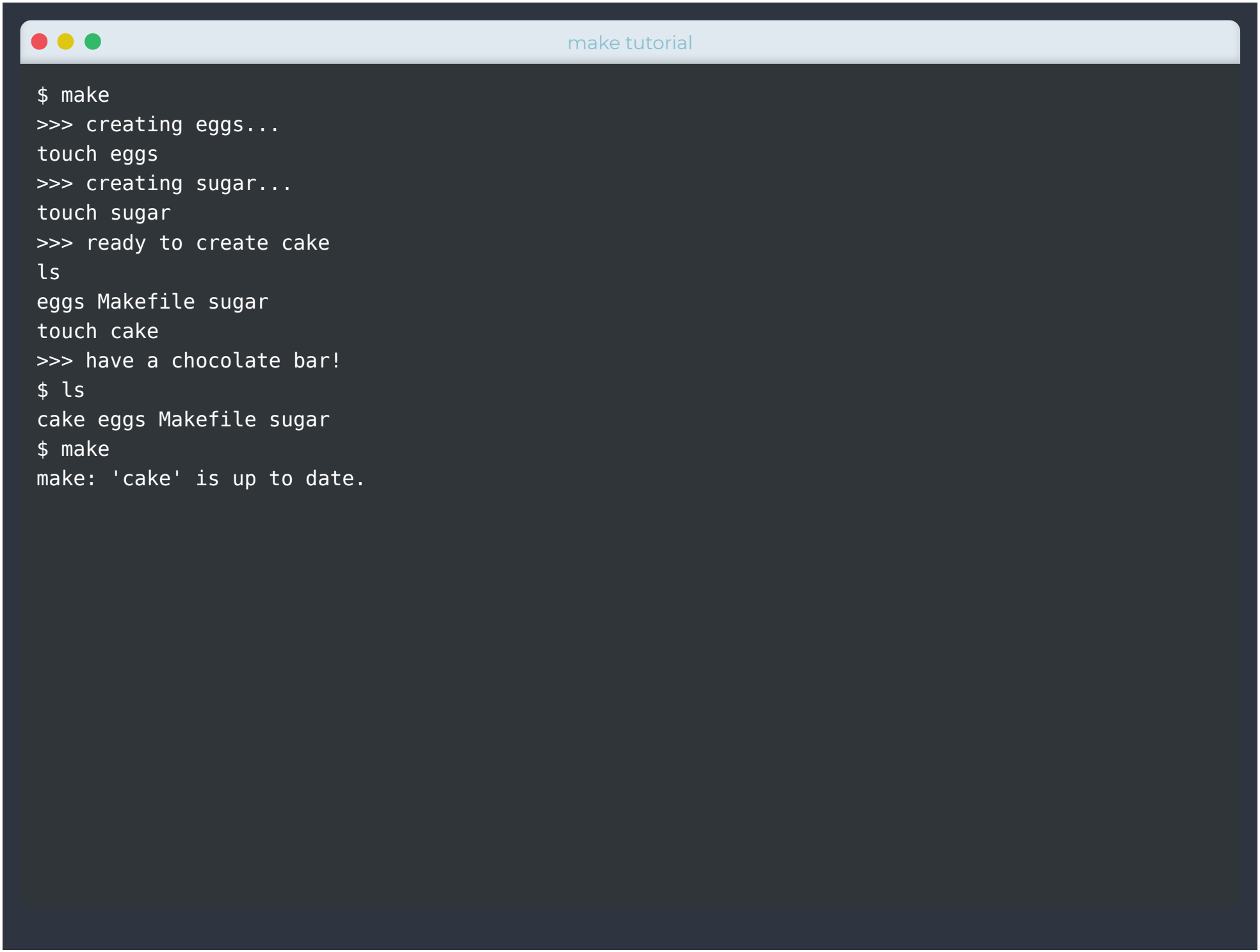
1. Simple
2. Versatile
3. Ubiquitous

Common use case:

creating artifacts (files) with the minimum amount of work

The anatomy of a Makefile

- Rules
- Macros (also used as variables)
- Comments
- Directives (conditionals, including other Makefiles, etc)


make tutorial
$ make
>>> creating eggs...
touch eggs
>>> creating sugar...
touch sugar
>>> ready to create cake
ls
eggs Makefile sugar
touch cake
>>> have a chocolate bar!
$ ls
cake eggs Makefile sugar
$ make
make: 'cake' is up to date.

Our case study

Hello function and accompanying header

```
<stdio.h>
"hello.h"

o(char *name)

e == NULL) name = "world";
"Hello, %s!\n", name);
```

```
#ifndef __HELLO_H
#define __HELLO_H

void hello(char *name);

#endif
```

Main file

```
#include <stdio.h>
#include "hello.h"

int main(int argc, char** argv)
{
  hello(NULL);
  // Greet the commandline arguments in reverse order
  while(argc-- > 1)
  {
    hello(argv[argc]);
  }
  return 0;
}
```

```
hello: main.o hello.o
	gcc -Wall main.o hello.o -o hello

main.o: main.c hello.h
	gcc -Wall -c main.c -o main.o

hello.o: hello.c hello.h
	gcc -Wall -c hello.c -o hello.o

clean:
	rm -f *.o hello
```

PREDEFINED VARIABLES

`$@` - Name of target
`$<` - The first prerequisite
`$^` - All prerequisites
`$?` - All newer prerequisites
... and more!

`$(CC)` - The C compiler (defaults to cc)
`$(CFLAGS)` - Flags for CC (empty by default)
`$(RM)` - The command used to remove files
... and more, mostly outdated!

NOW GO make A DIFFERENCE

You have the power