Command Tube

Automated Yourself

Michael Han

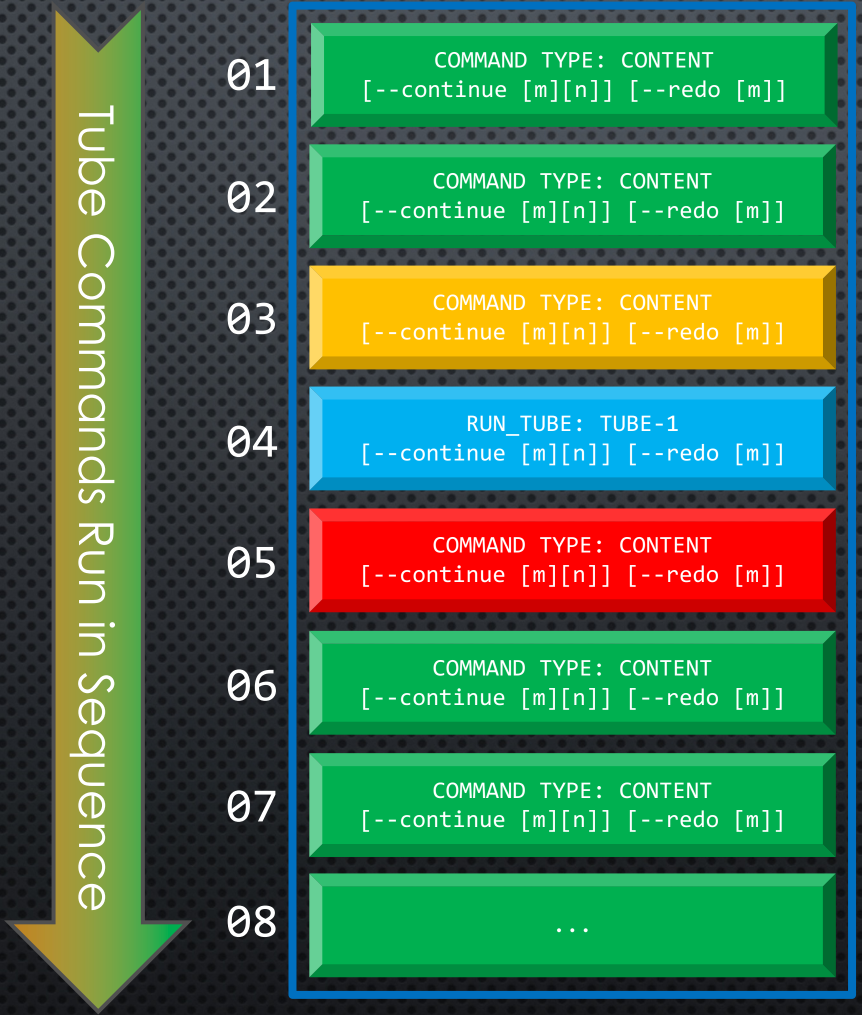
2022

Command Tube

Command Tube can run a group of sequenced commands

Each command has general arguments and self arguments

Easy access bult-in help document



Tube Commands Run in Sequence

01 COMMAND TYPE: CONTENT
[--continue [m][n]] [--redo [m]]

02 COMMAND TYPE: CONTENT
[--continue [m][n]] [--redo [m]]

03 COMMAND TYPE: CONTENT
[--continue [m][n]] [--redo [m]]

04 RUN_TUBE: TUBE-1
[--continue [m][n]] [--redo [m]]

05 COMMAND TYPE: CONTENT
[--continue [m][n]] [--redo [m]]

06 COMMAND TYPE: CONTENT
[--continue [m][n]] [--redo [m]]

07 COMMAND TYPE: CONTENT
[--continue [m][n]] [--redo [m]]

08 ...

Support 55+ command types for current

>>> tube help commands

🟩 Command run successfully.

🟨 Command is skipped.

🟥 Command run failed.

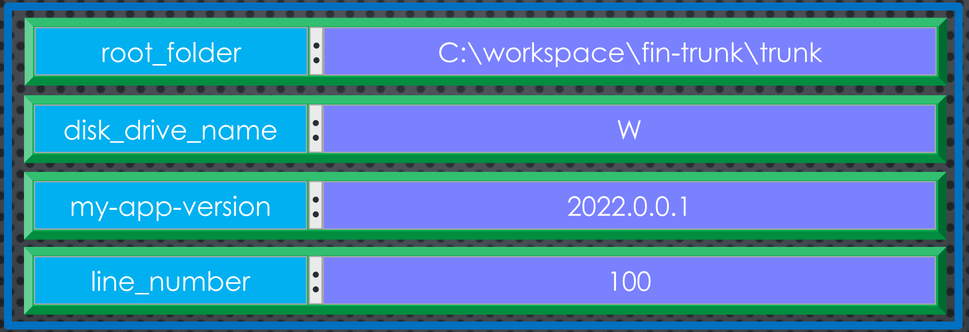
https://github.com/michael-hll/command-tube

Tube Variables

Define Variables

- Manually added in the tube file
- Use SET tube command
- Updated via some tube commands

Variables



root_folder	C:\workspace\fin-trunk\trunk
disk_drive_name	W
my-app-version	2022.0.0.1
line_number	100

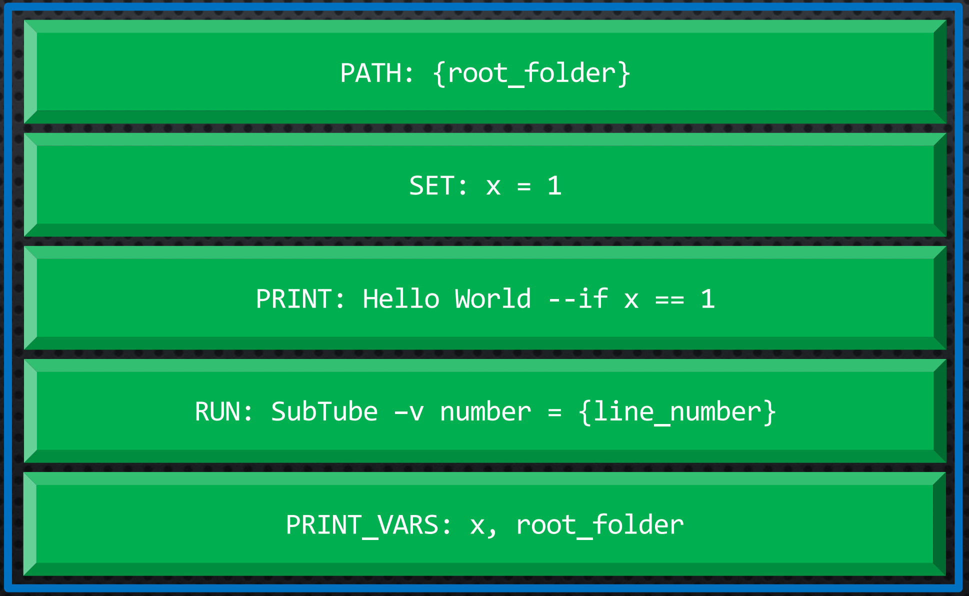
Pass Variables

- From terminal, use syntax ‘-v var1=value1, var2=value2’ to pass variables from console to tube
- Pass variable to sub tube when use RUN tube command

Use Variables

- Tube command content can use syntax {variable-name} as the placeholder
- In --if condition or --while condition and set tube command, you can omit the curly brackets

Tube



```
PATH: {root_folder}
SET: x = 1
PRINT: Hello World --if x == 1
RUN: SubTube -v number = {line_number}
PRINT_VARS: x, root_folder
```

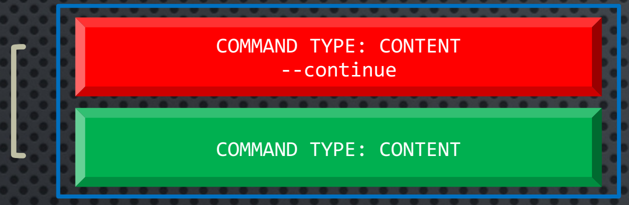
View Variables

- Use PRINT_VARS tube command
- From terminal, use syntax ‘help vars’ to view all variables in console

continue & redo

General Arguments I

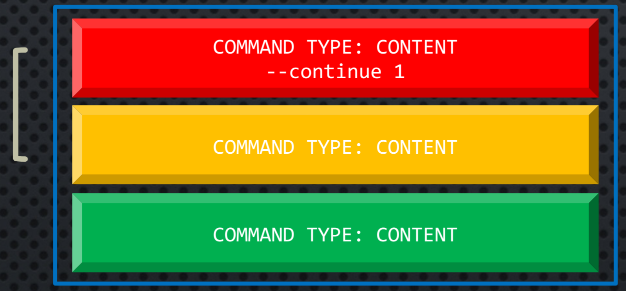
continue



COMMAND TYPE: CONTENT
--continue

COMMAND TYPE: CONTENT

continue [m]

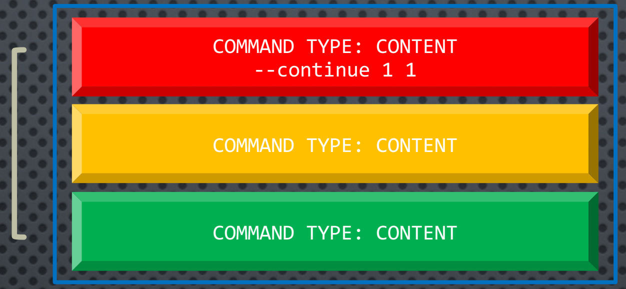


COMMAND TYPE: CONTENT
--continue 1

COMMAND TYPE: CONTENT

COMMAND TYPE: CONTENT

continue [m] [n]

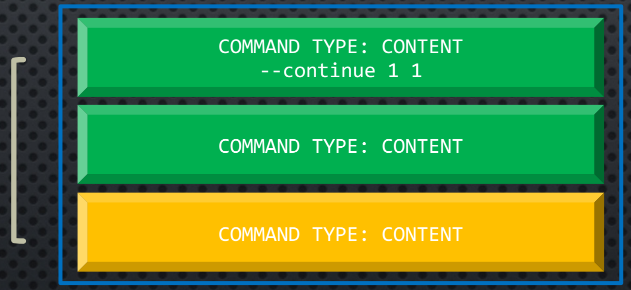


COMMAND TYPE: CONTENT
--continue 1 1

COMMAND TYPE: CONTENT

COMMAND TYPE: CONTENT

continue [m] [n]

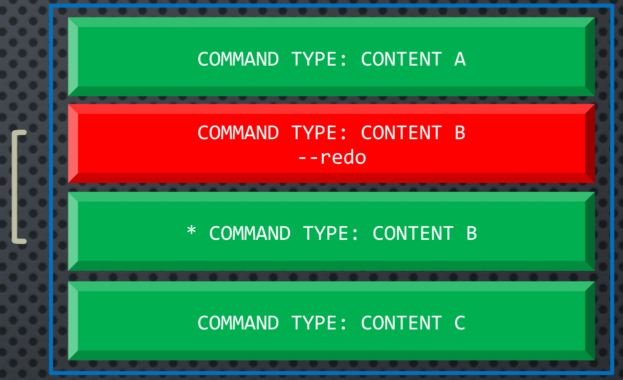


COMMAND TYPE: CONTENT
--continue 1 1

COMMAND TYPE: CONTENT

COMMAND TYPE: CONTENT

redo



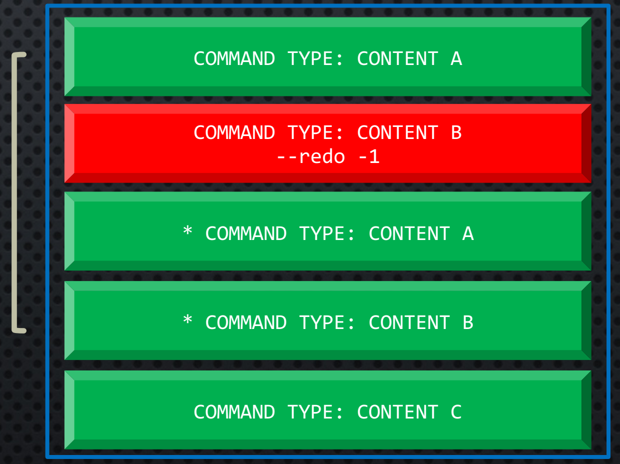
COMMAND TYPE: CONTENT A

COMMAND TYPE: CONTENT B
--redo

* COMMAND TYPE: CONTENT B

COMMAND TYPE: CONTENT C

redo [m]



COMMAND TYPE: CONTENT A

COMMAND TYPE: CONTENT B
--redo -1

* COMMAND TYPE: CONTENT A

* COMMAND TYPE: CONTENT B

COMMAND TYPE: CONTENT C

--continue: Looks like IF..ELSE..

Decides which command will be skipped and which command executed, based on command execute result (SUCCESS/FAIL).
Syntax: --continue [m] [n]

--redo: Redo your command

Redo fail command or repeat success command multiple times
Syntax: --redo [m]

🟩 Command run successfully.
🟨 Command is skipped.
🟥 Command run failed.

if condition

General Arguments II

Variables

not_run	:	no
run_command	:	yes
run-type	:	server
s or S	:	SPACE-CHAR

Tube

```
PATH: C:\users --if not_run == no
```

```
PATH: C:\dev --if run_command
```

```
PATH: C:\dev --if not_run run_command
```

```
PATH: C:\dev --if “server”==run-type
```

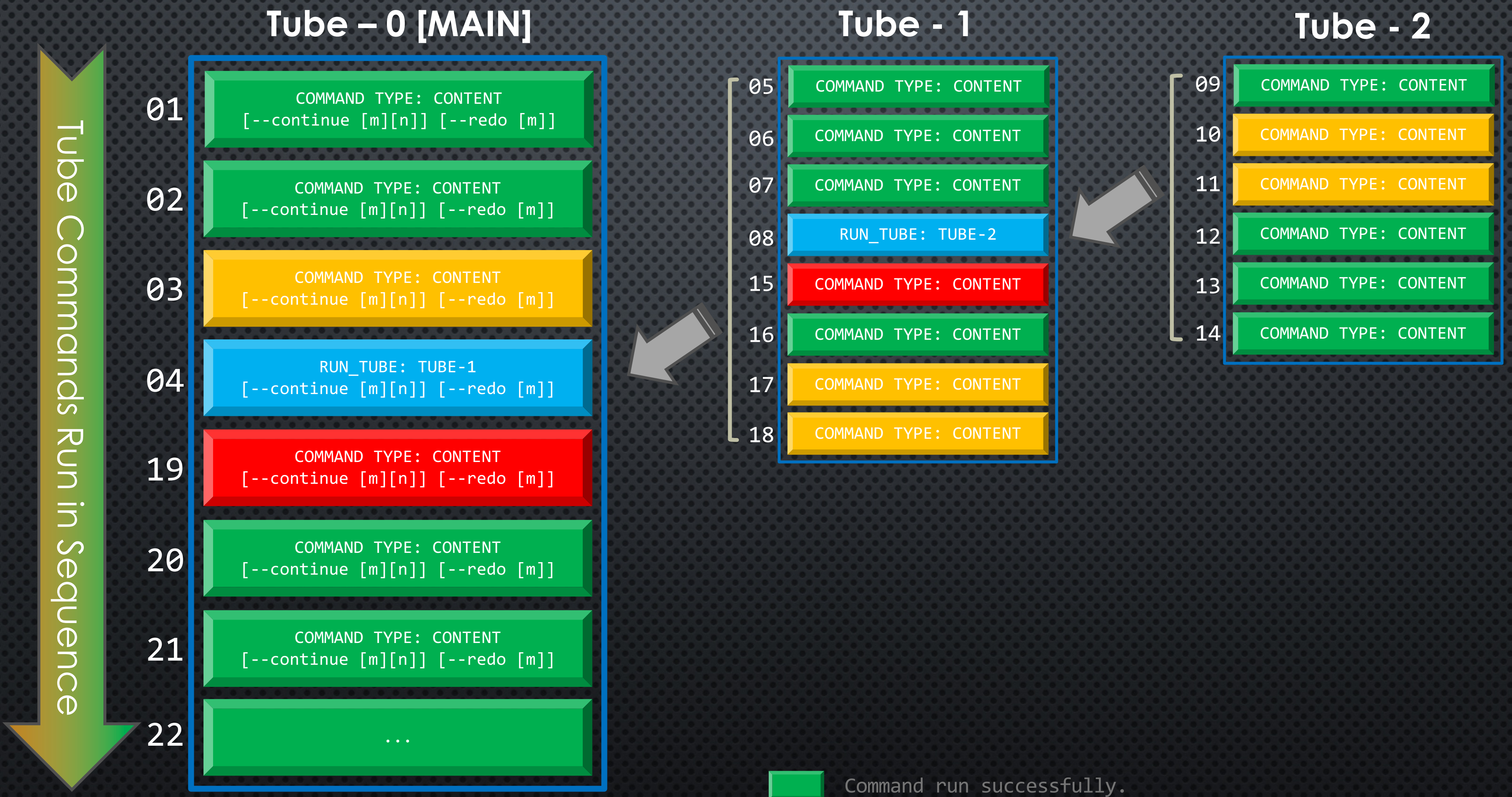
--if: Check if run a command

The {variable-name} will be replaced by the variable value. Only the following values will skip the running: ‘false’, ‘False’, ‘No’, ‘no’

Syntax: --if <condition>

For more general arguments please refer to README

Sub Tubes



Tube – 0 [MAIN]

01 COMMAND TYPE: CONTENT [--continue [m][n]] [--redo [m]]

02 COMMAND TYPE: CONTENT [--continue [m][n]] [--redo [m]]

03 COMMAND TYPE: CONTENT [--continue [m][n]] [--redo [m]]

04 RUN_TUBE: TUBE-1 [--continue [m][n]] [--redo [m]]

19 COMMAND TYPE: CONTENT [--continue [m][n]] [--redo [m]]

20 COMMAND TYPE: CONTENT [--continue [m][n]] [--redo [m]]

21 COMMAND TYPE: CONTENT [--continue [m][n]] [--redo [m]]

22 ...

Tube - 1

05 COMMAND TYPE: CONTENT

06 COMMAND TYPE: CONTENT

07 COMMAND TYPE: CONTENT

08 RUN_TUBE: TUBE-2

15 COMMAND TYPE: CONTENT

16 COMMAND TYPE: CONTENT

17 COMMAND TYPE: CONTENT

18 COMMAND TYPE: CONTENT

Tube - 2

09 COMMAND TYPE: CONTENT

10 COMMAND TYPE: CONTENT

11 COMMAND TYPE: CONTENT

12 COMMAND TYPE: CONTENT

13 COMMAND TYPE: CONTENT

14 COMMAND TYPE: CONTENT

🟩 Command run successfully.

🟨 Command is skipped.

🟥 Command run failed.

Tube Examples

e.g. Hello World

```
Tube:
    - print: Hello World
```

e.g. If condition

```
Variables:
    x: 1
Tube:
    - print: Hello World --if x == 1
```

e.g. Run Sub Tube

```
Tube:
    - print: output from main tube
    - run: SubTube
SubTube:
    - print: output from sub tube
```

e.g. Loop

```
Tube:
    - set: ls = ["hello", "world"]
    - run: SubTube --each i, item in ls
SubTube:
    - print: >
        I’m in loop: {i}, value: {item}
```

Run Command Tube

A: Run command tube via source code (SRC mode)

Run at once:

```
>>> python command-tube.py –t tube.yaml -f
```

Run at next 6:00 o’clock:

```
>>> python command-tube.py –t tube.yaml --datetime n6
```

Run at 6:00 for 7 days:

```
>>> python command-tube.py –t tube.yaml --datetime n6 –l 1d --times 7
```

Show arguments usage:

```
>>> python command-tube.py –h
```

```
>>> ...
```

B: Run programming tube via package (BIN mode)

Run at once:

```
>>> tube –t tube.yaml -f
```

```
>>> ...
```