

Concurrency Patterns

NaiveDateTime

01

About


02

Managing Goroutines

03

Generators

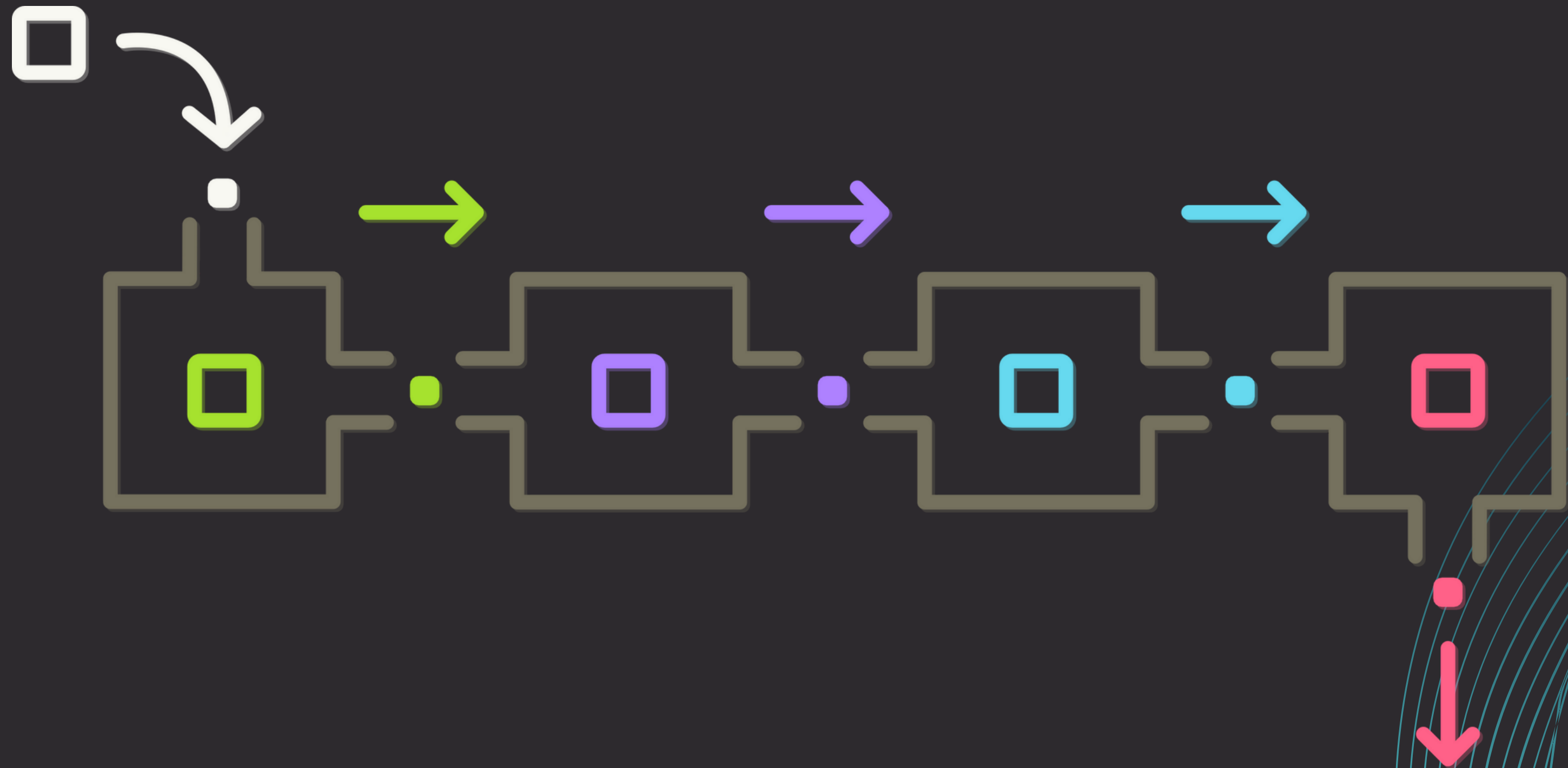
About

- | Channels and goroutines must be cleaned up at some point
 - | Failing to do so is a resource leak
 - | Concurrency patterns offer ways to clean up goroutines
 - | Also help with managing data flow
 - | Multiple patterns available based on the situation
- 
- An abstract graphic consisting of several thin, curved, teal-colored lines that sweep across the right side of the slide, creating a sense of motion and depth.

Pipelines

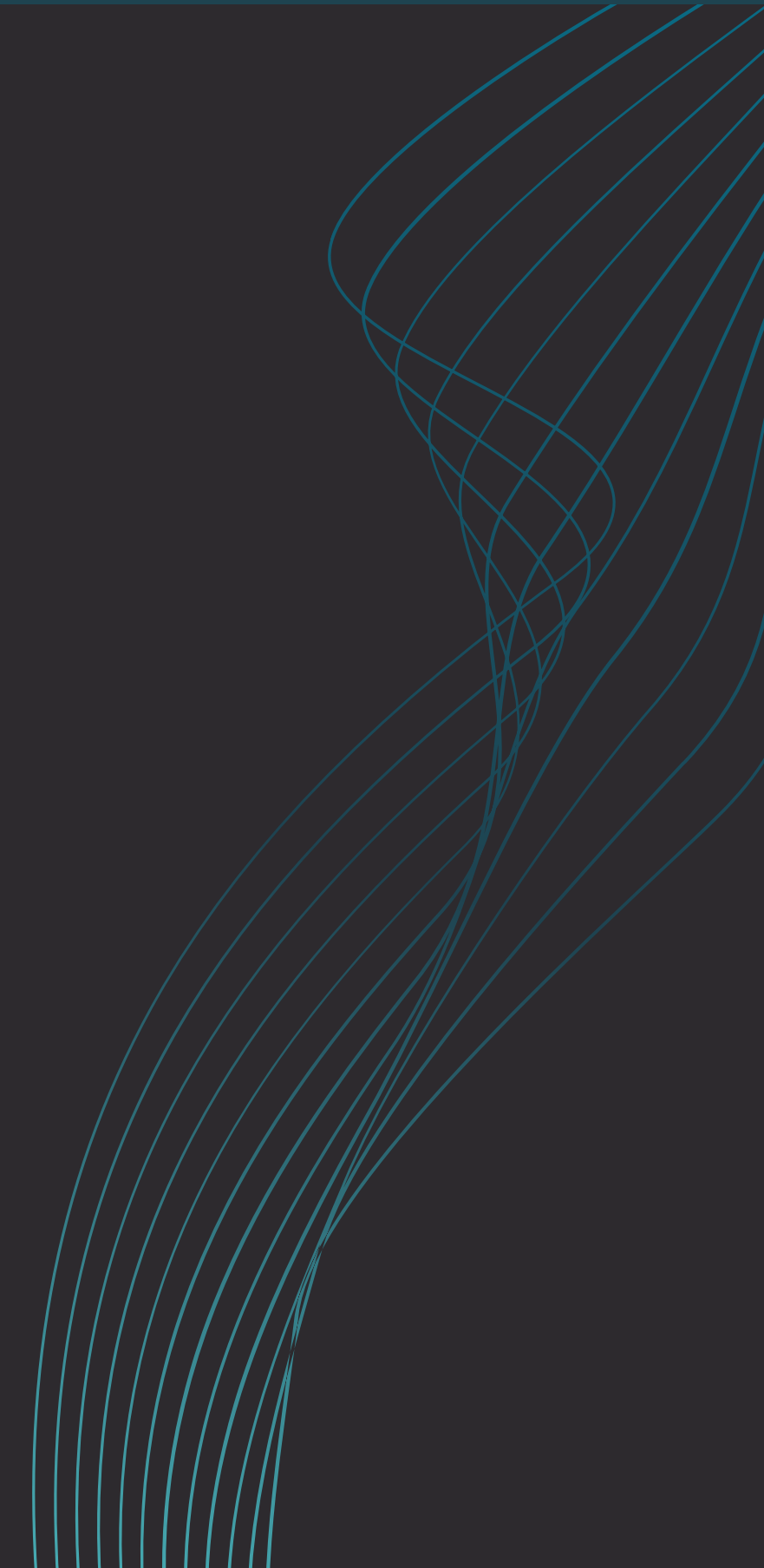
- | Pipelines are multiple stages of operations connected using channels
 - | At least 1 input channel and 1 output channel
- | Use goroutines to manage the sending and receiving of values
- | Multiple options for managing and combining stages
- | Each stage of the pipeline depends on the previous stage

Pipeline - Visual

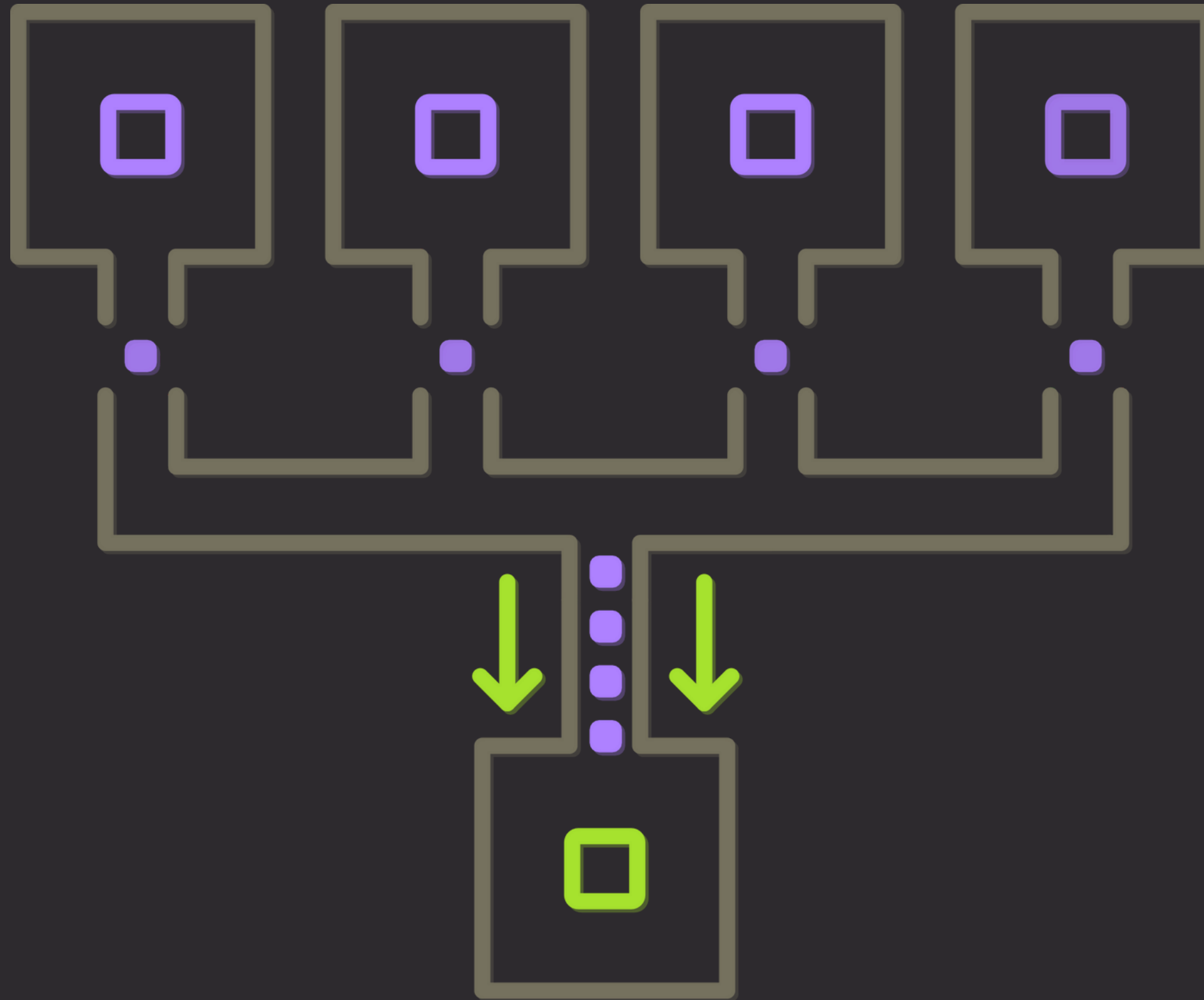


Fan-in

- | Multiple input channels, 1 output channel
- | Examples:
 - | Serializing requests to purchase an item
 - | Segmented processing of a single item



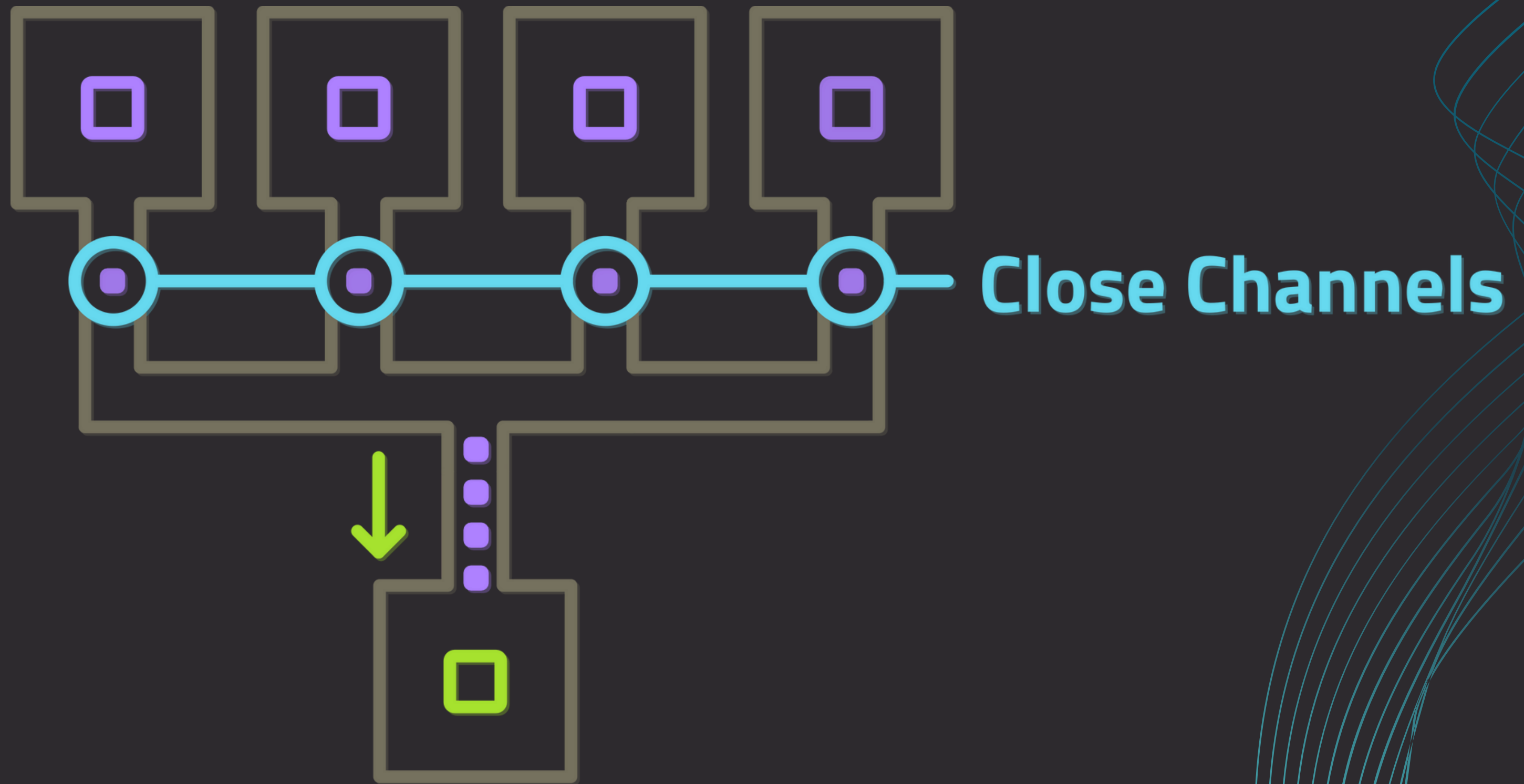
Fan-in



Close / Cancellation

- | Closing channel indicates end of data stream
- | Pipeline stage should close channel when work is done
 - | Cleanup goroutines / no resource leaks / unblock

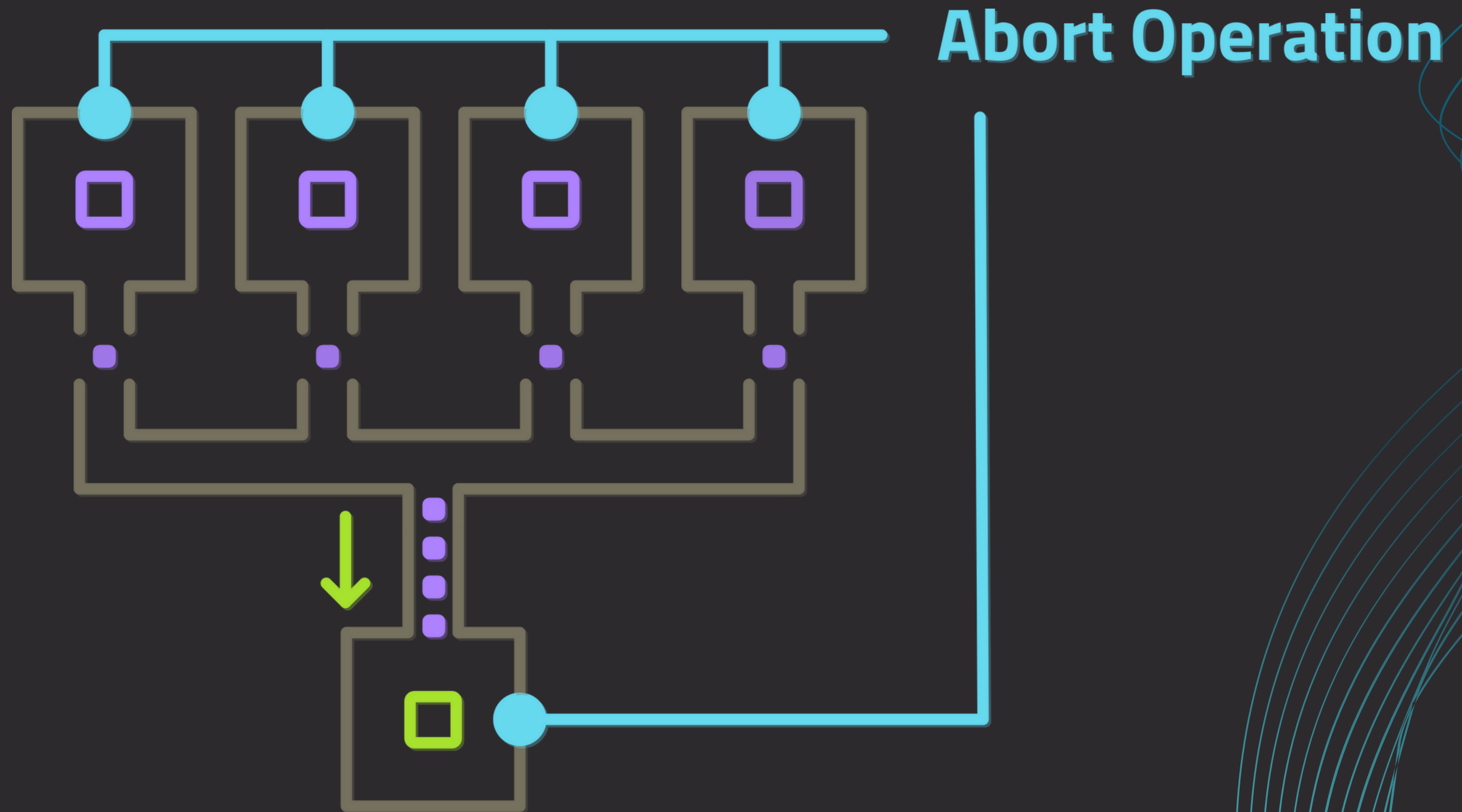
Close / Cancellation



Request Quit

- | Dedicated "quit" channel
- | Stage listens on incoming "data" channel and "quit" channel
 - | If data/signal comes in on "quit" channel, goroutine shuts down
- | Can use one or multiple "quit" channels
 - | Multiple: Must send enough signals for each stage to quit

Request Quit

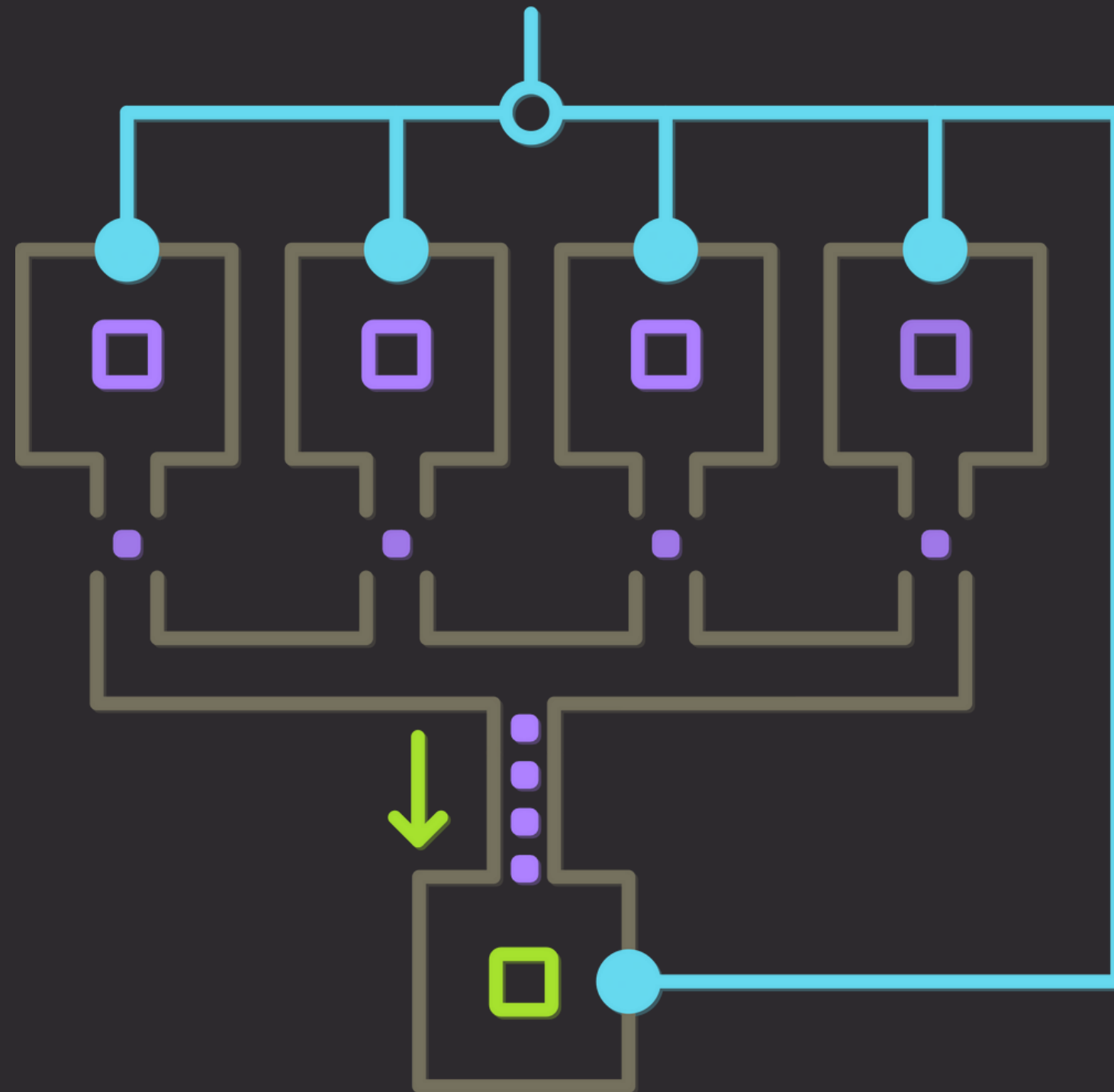


Context

- | Similar to "request quit"
- | No need to manually manage channels
- | Calling a "quit" function cancels all operations using the Context
- | Can associate data with each Context
 - | IP addresses, session IDs, node identifiers, etc

Context

Abort Operation



Generator

- | On-demand yielding of items
- | Items generated until bounded channel fills
 - | Reduced processing
- | Whenever items are read from the channel, new ones are calculated as needed

Generator

Write / Send

Msg ■



Read / Receive

■ **Msg**



Recap

- | Concurrency patterns are used to manage goroutines
- | Pipelines are broken into "stages" with a goroutine for each stage
 - | Stages communicate with channels
- | Multiple ways to clean up pipelines:
 - | Close channels / send "quit" signal / use Context
- | Generators can be used to yield items on-demand

Heading

- | Lorem ipsum dolor sit amet, consectetur adipiscing elit,
- | Lorem ipsum dolor sit amet, consectetur adipiscing elit,
 - | Lorem ipsum dolor sit amet, consectetur adipiscing elit,
- | Lorem ipsum dolor sit amet, consectetur adipiscing elit,

