

plumber + future: Async Web APIs

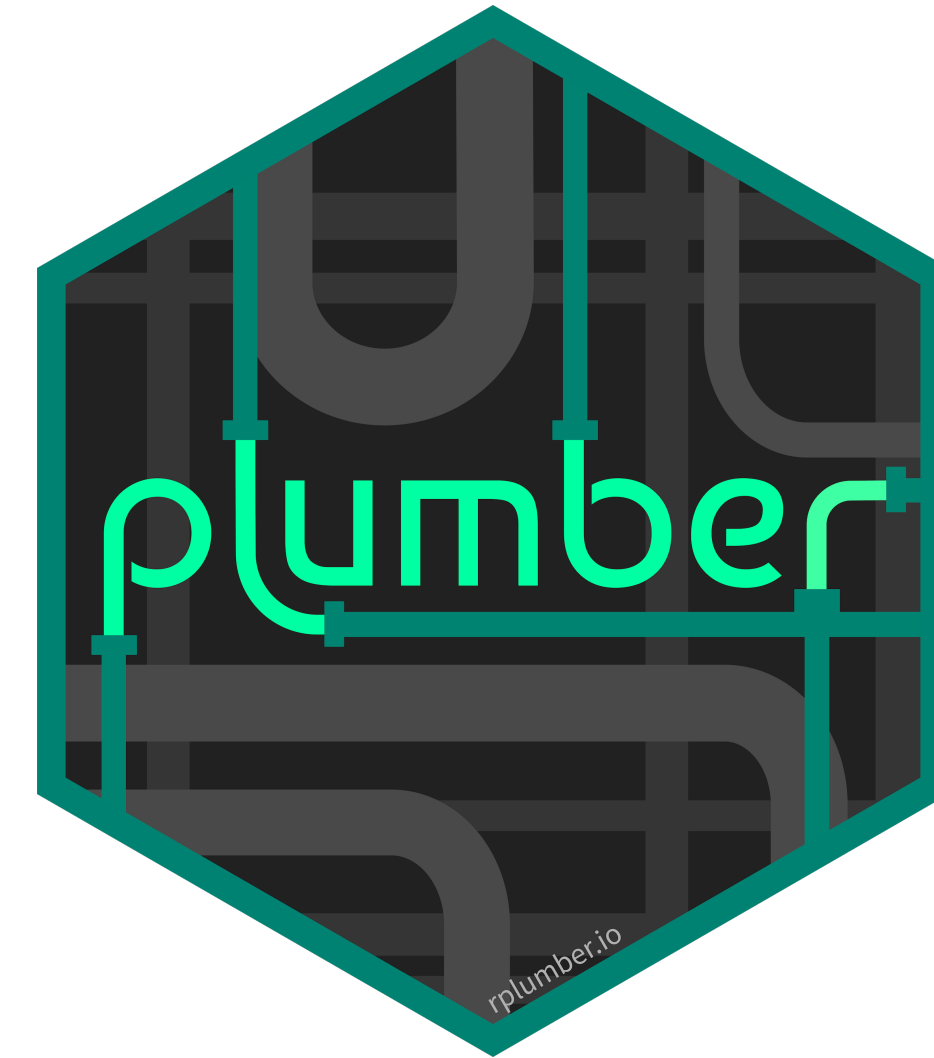

```
function(id) {  
    fast_calc(id)  
}
```

```
function(id) {  
    slow_calc(id)  
}
```

```
# ./plumber.R
```

```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

```
## @get /slow/<id>  
function(id) {  
  slow_calc(id)  
}
```





Receive



Process



Respond





Receive



Process



Respond





Receive

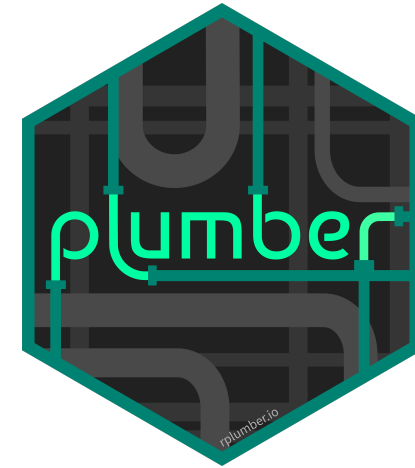


Process



Respond





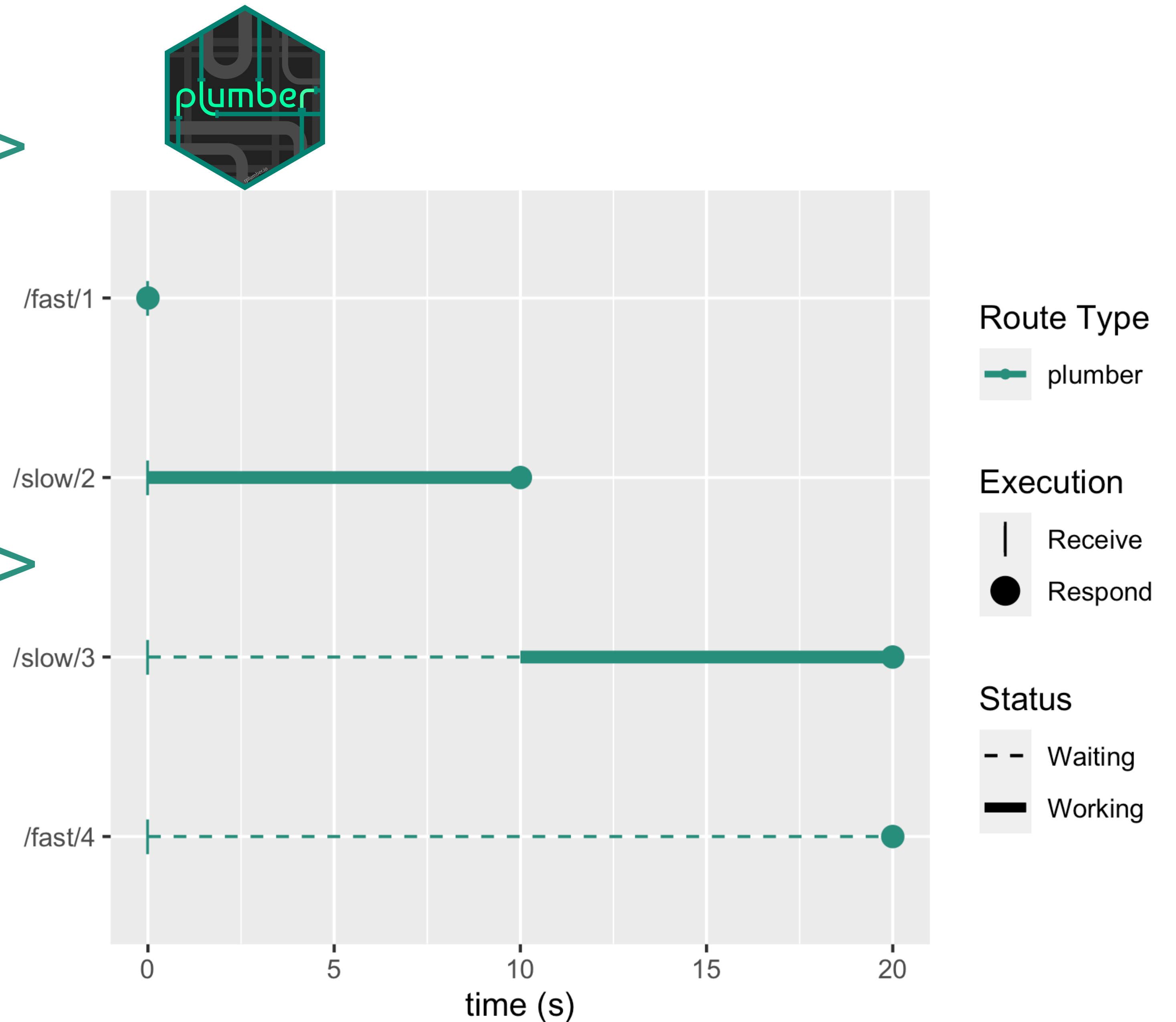
```
#* @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

```
#* @get /slow/<id>  
function(id) {  
  slow_calc(id)  
}
```



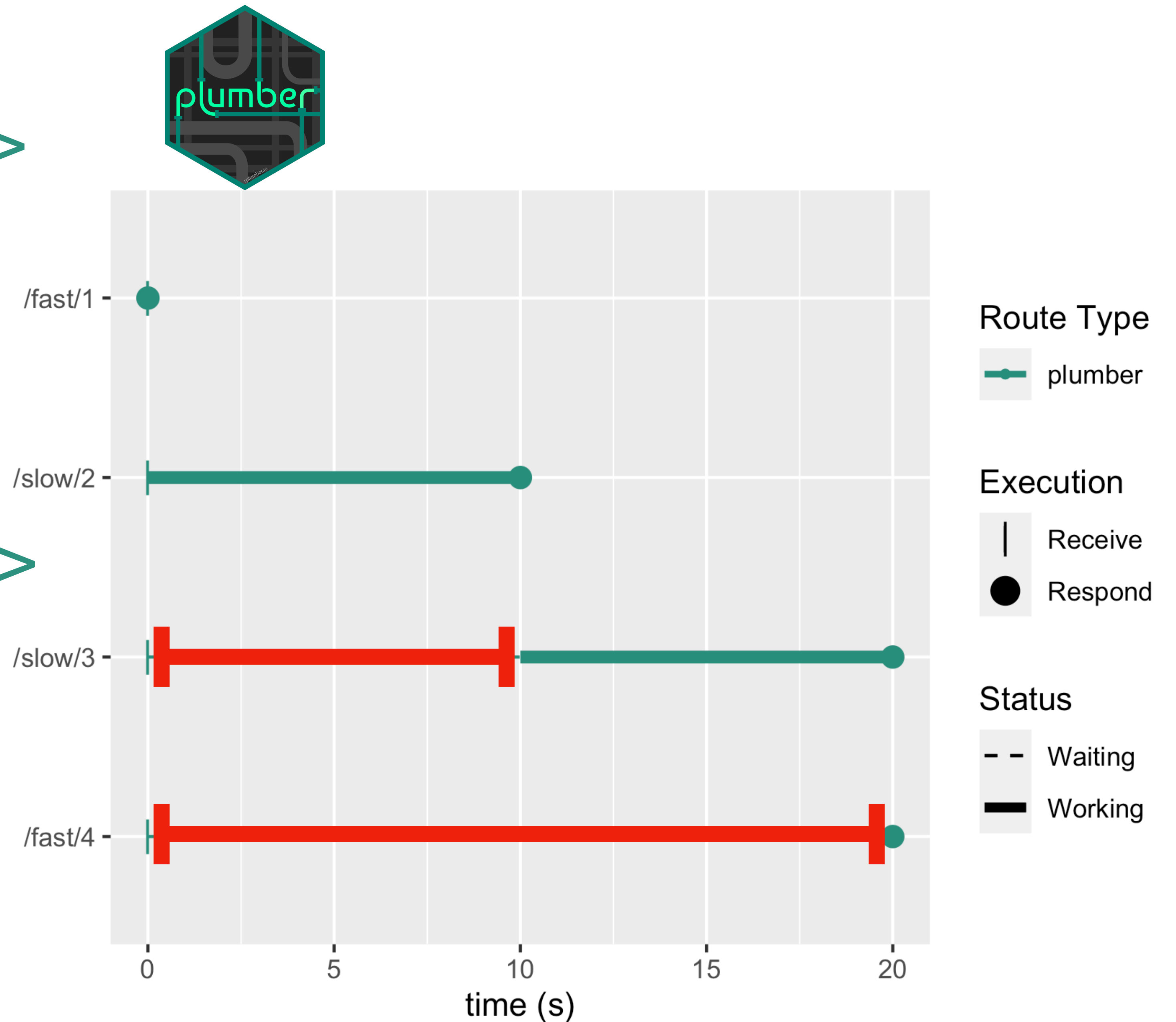
```
## @get /fast/<id>
function(id) {
  fast_calc(id)
}
```

```
## @get /slow/<id>
function(id) {
  slow_calc(id)
}
```



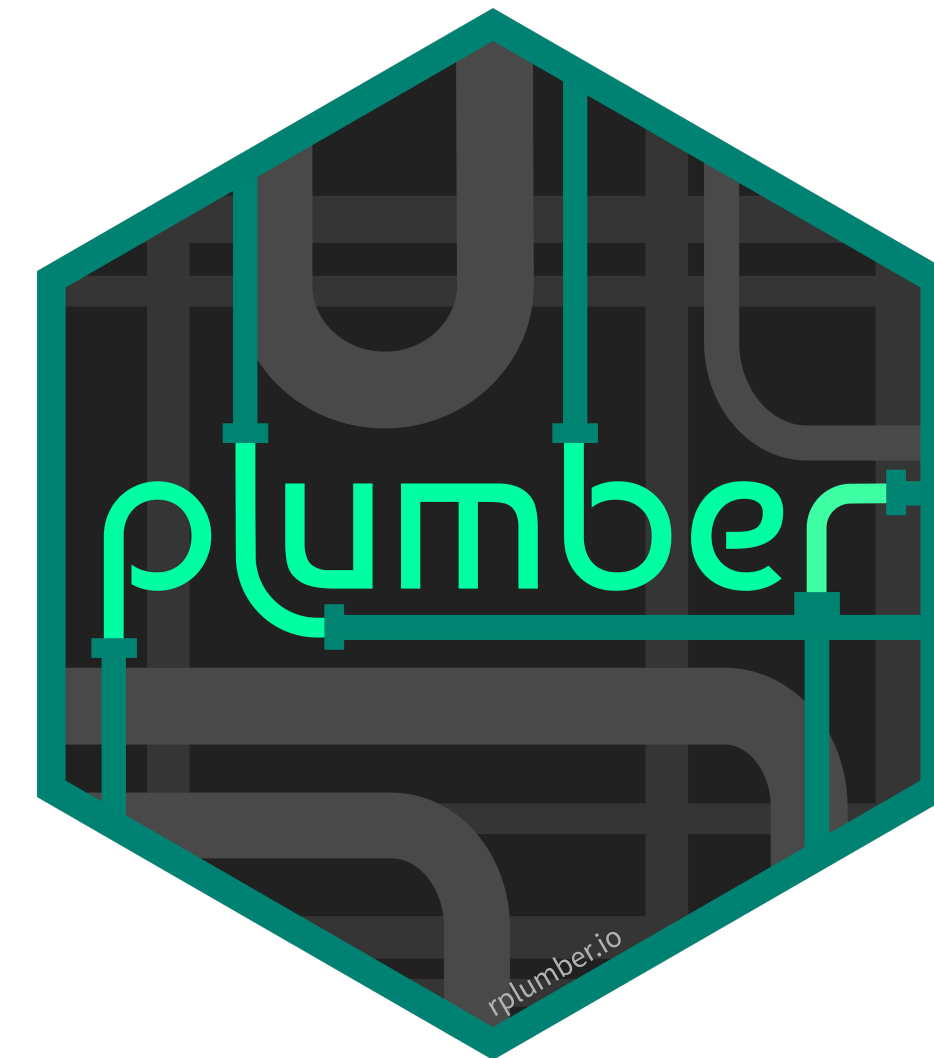
```
## @get /fast/<id>
function(id) {
  fast_calc(id)
}
```

```
## @get /slow/<id>
function(id) {
  slow_calc(id)
}
```



```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

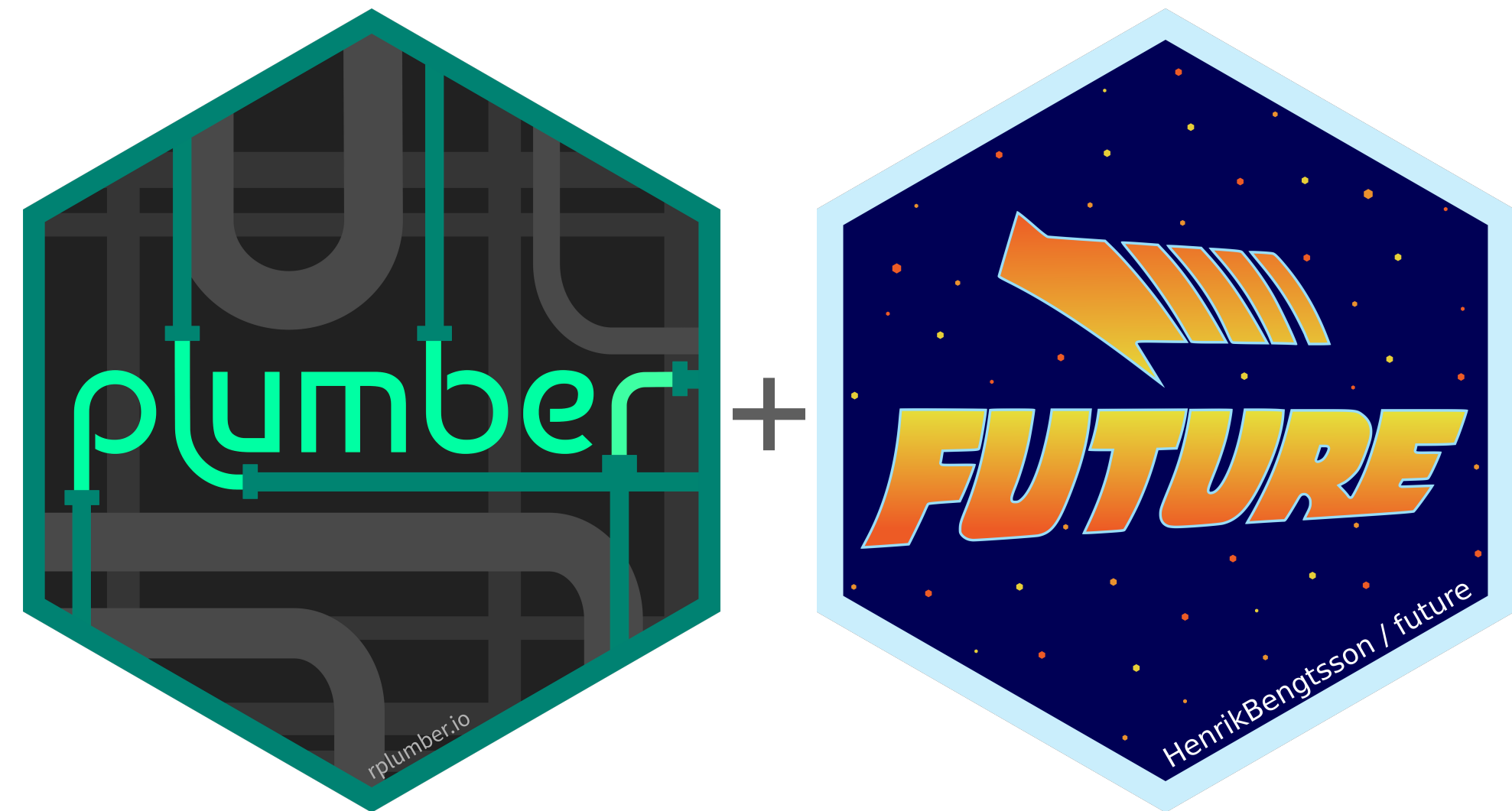
```
## @get /slow/<id>  
function(id) {  
  slow_calc(id)  
}
```



```
future::plan("multisession")
```

```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

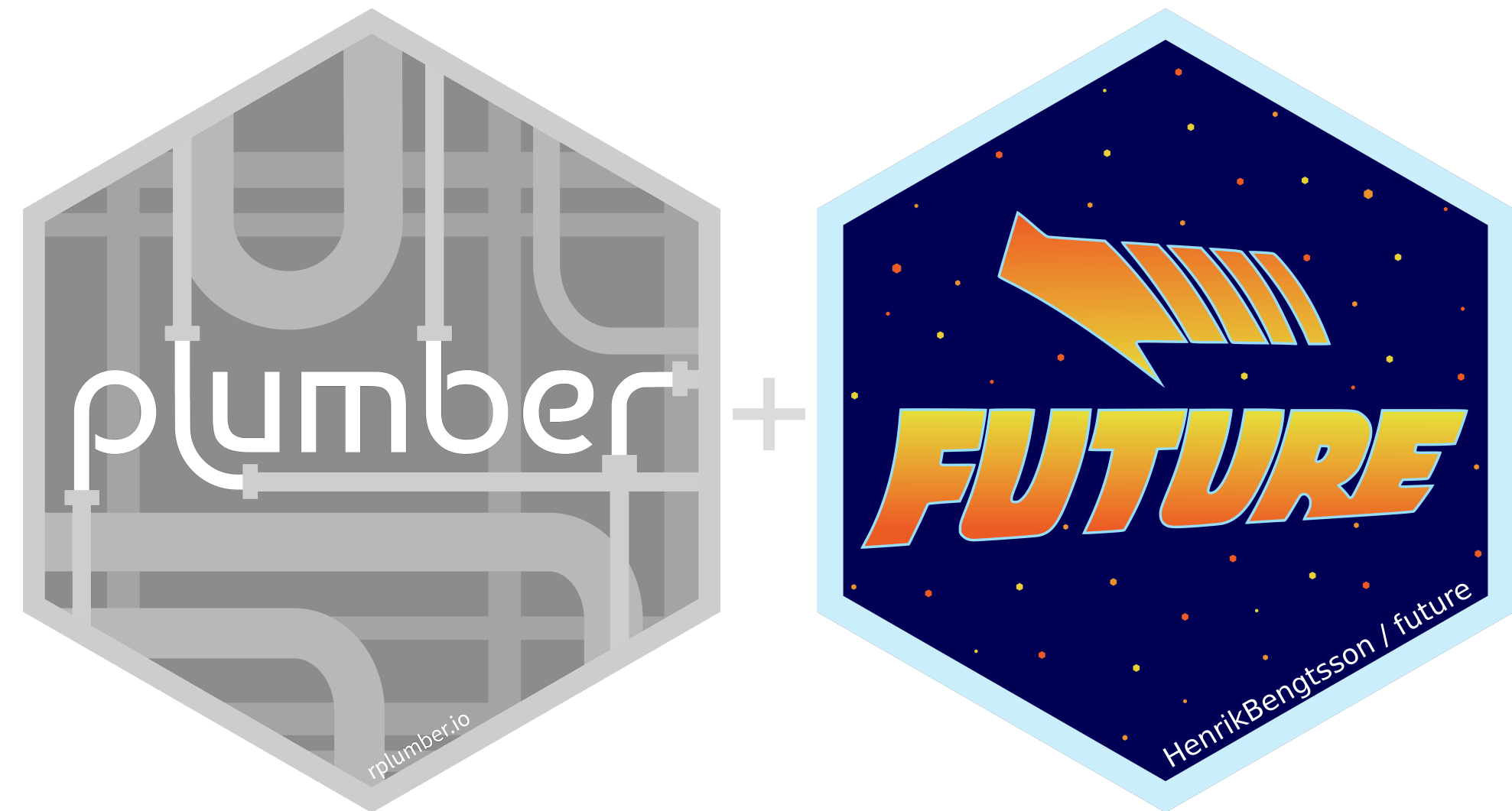
```
## @get /slow/<id>  
function(id) {  
  future::future({  
    slow_calc(id)  
  })  
}
```



```
future::plan("multisession")
```

```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

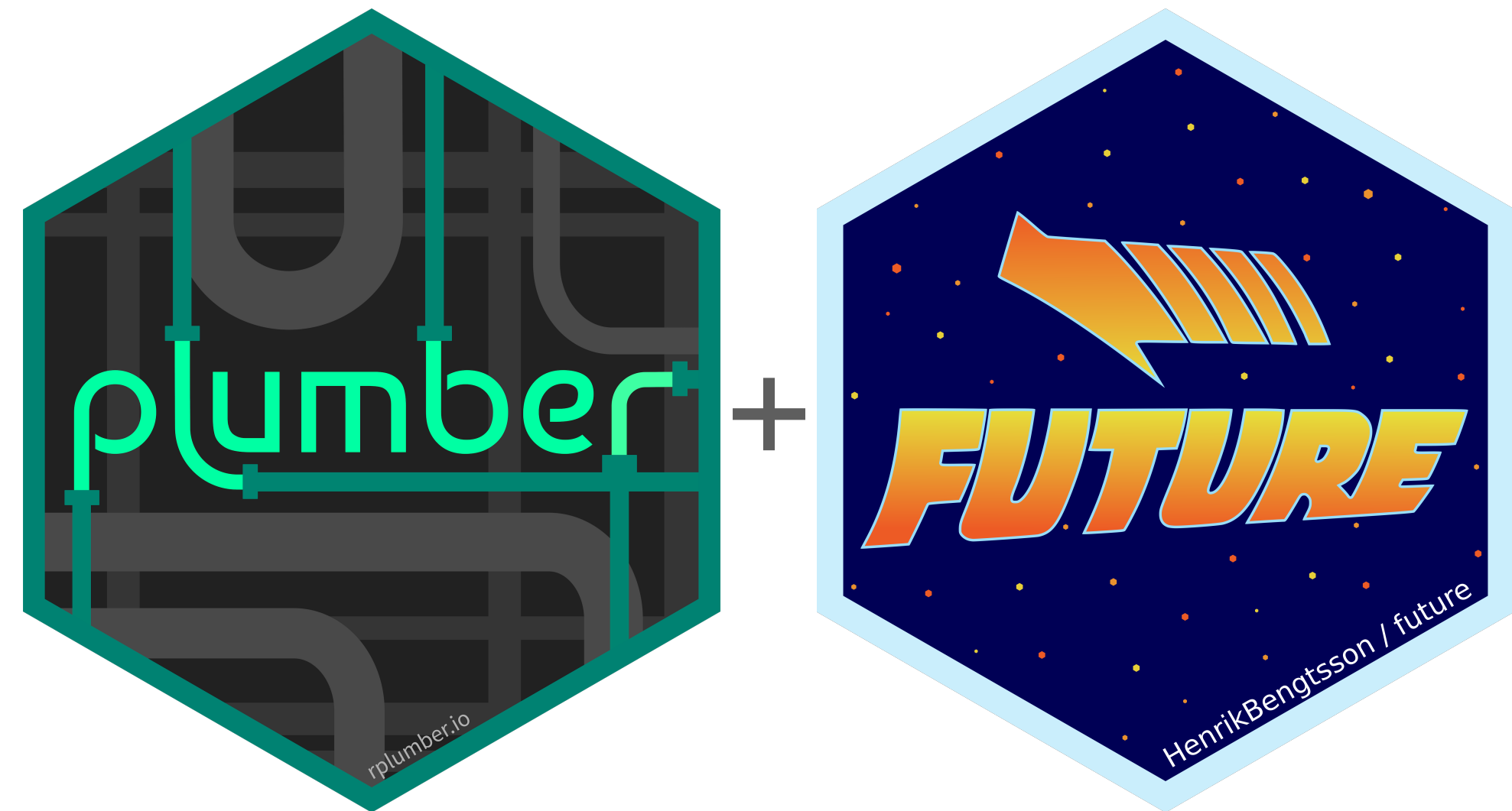
```
## @get /slow/<id>  
function(id) {  
  future::future({  
    slow_calc(id)  
  })  
}
```



```
future::plan("multisession")
```

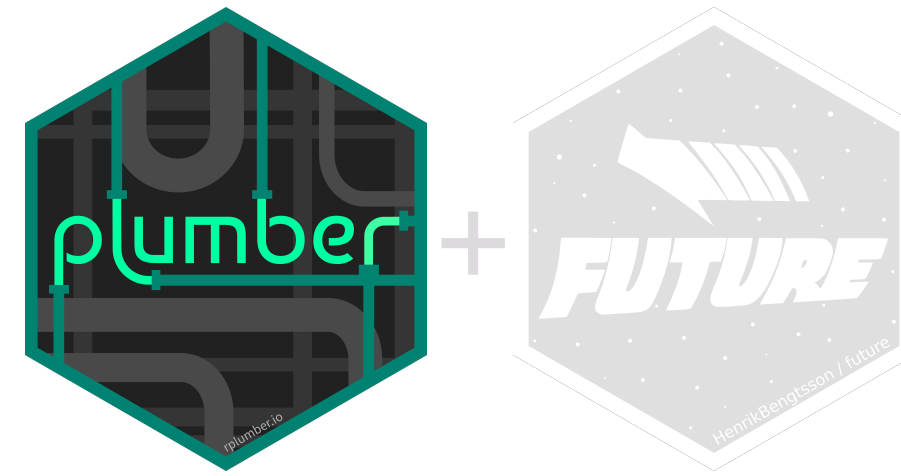
```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

```
## @get /slow/<id>  
function(id) {  
  future::future({  
    slow_calc(id)  
  })  
}
```



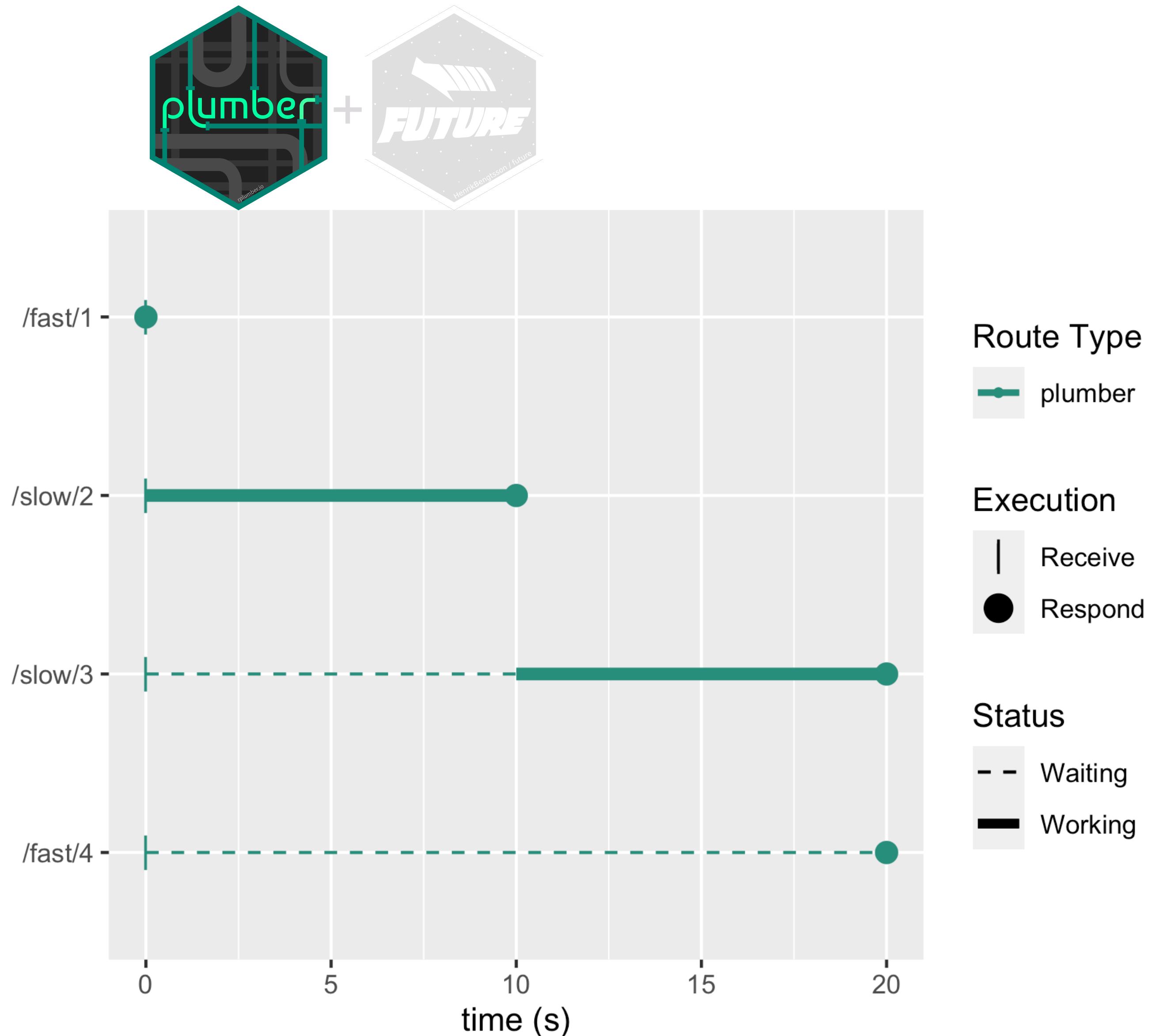
```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

```
## @get /slow/<id>  
function(id) {  
  
  slow_calc(id)  
  
}
```



```
## @get /fast/<id>
function(id) {
  fast_calc(id)
}
```

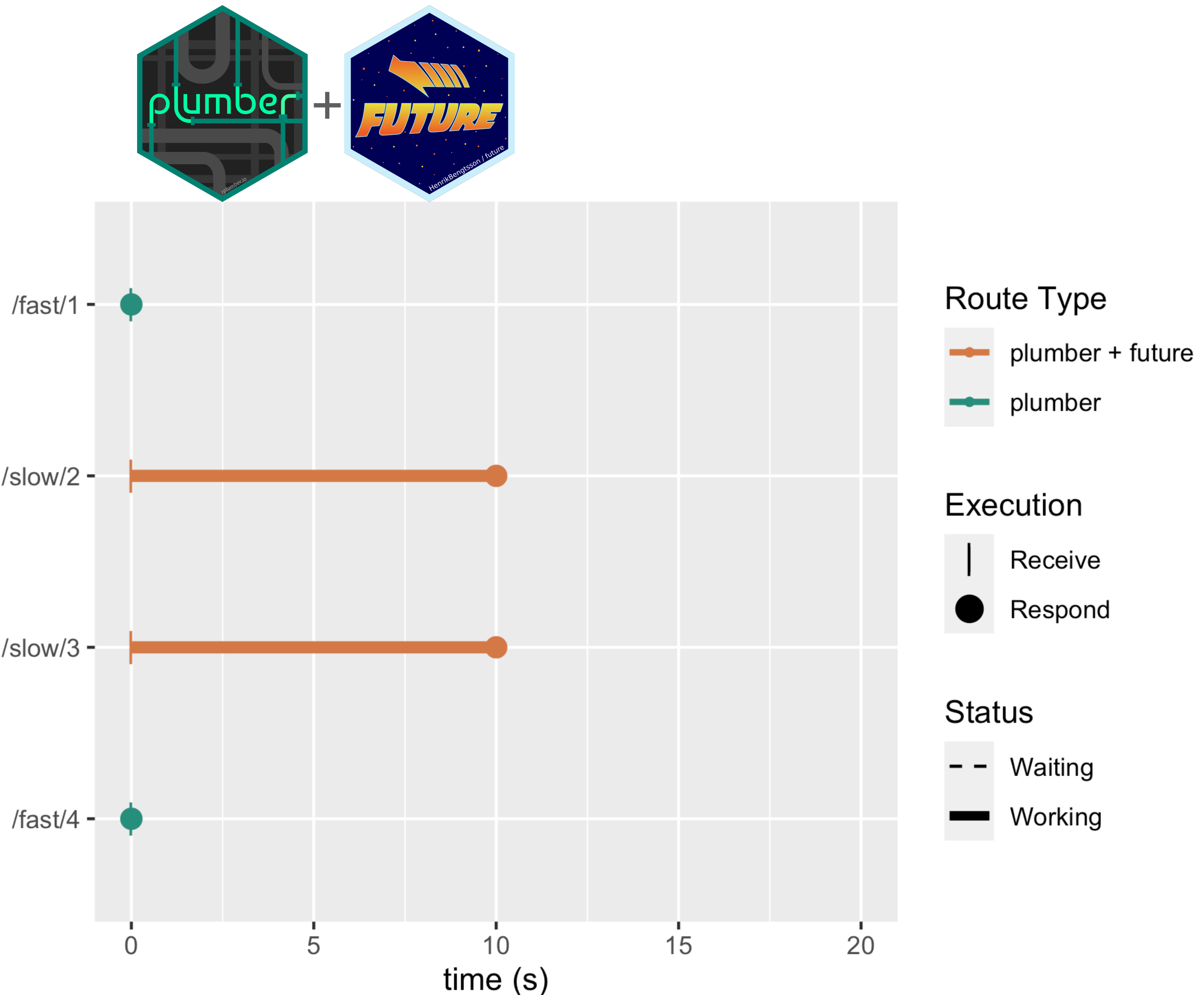
```
## @get /slow/<id>
function(id) {
  slow_calc(id)
}
```




```
future::plan("multisession")
```

```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

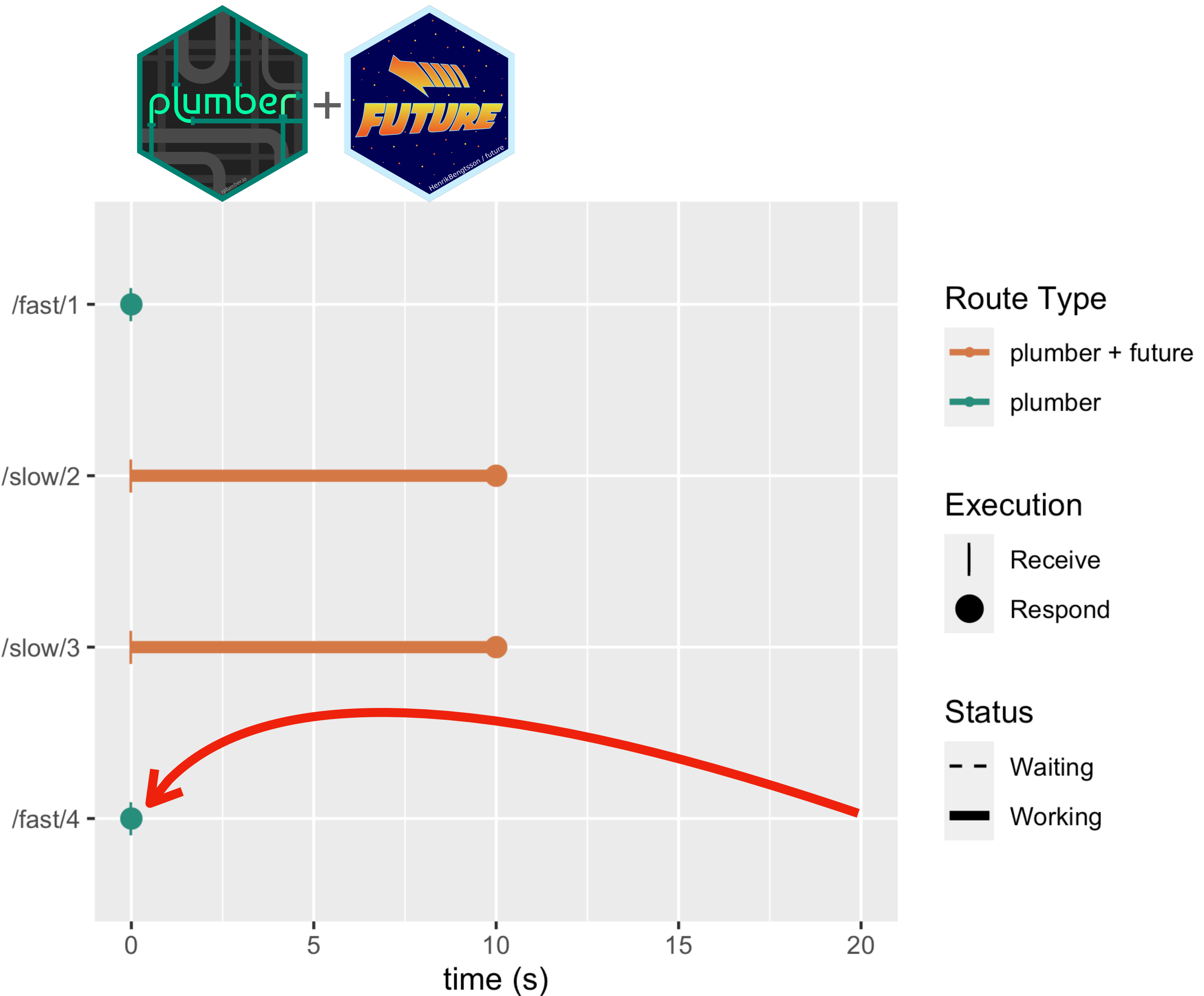
```
## @get /slow/<id>  
function(id) {  
  future::future({  
    slow_calc(id)  
  })  
}
```



```
future::plan("multisession")
```

```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

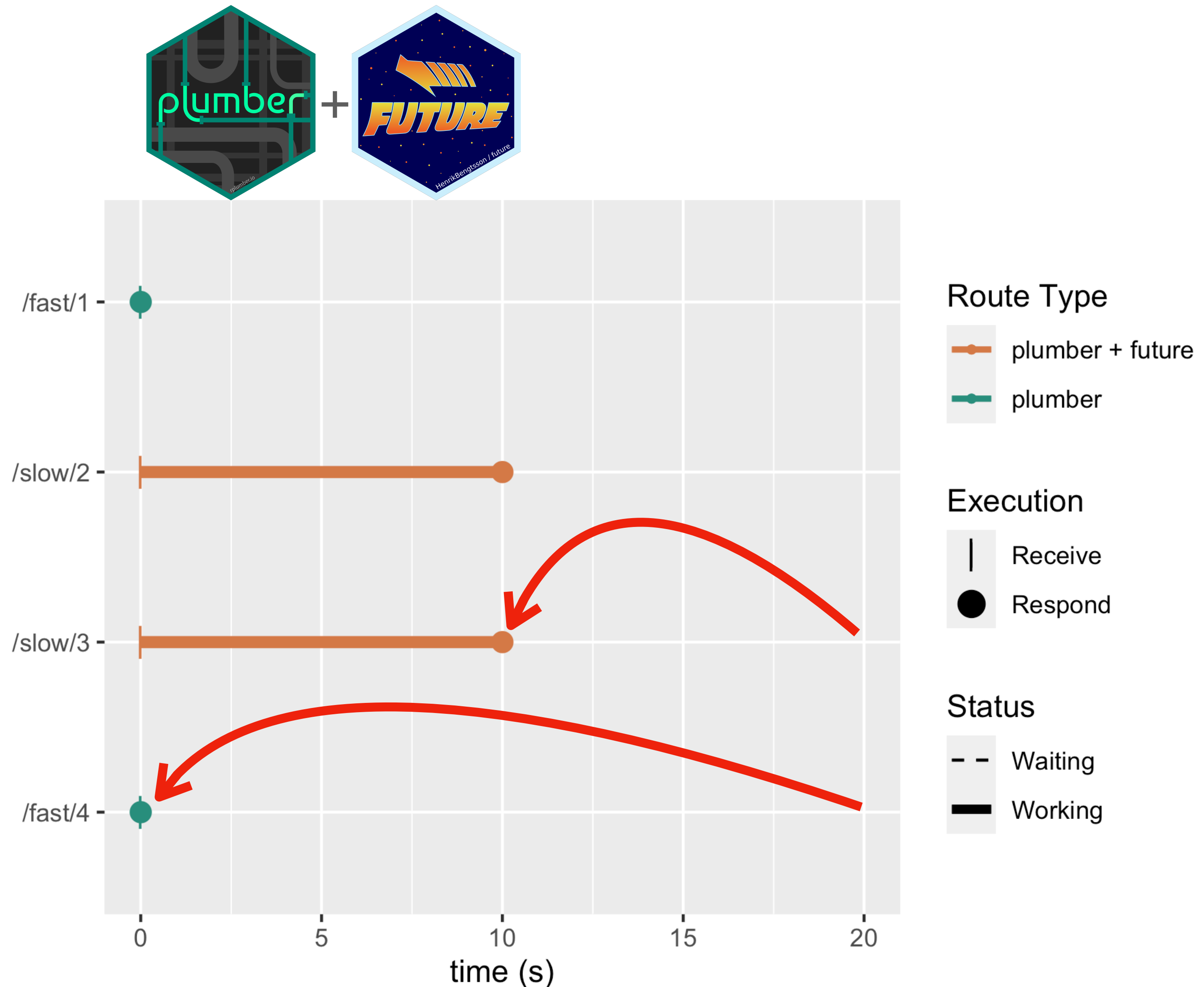
```
## @get /slow/<id>  
function(id) {  
  future::future({  
    slow_calc(id)  
  })  
}
```

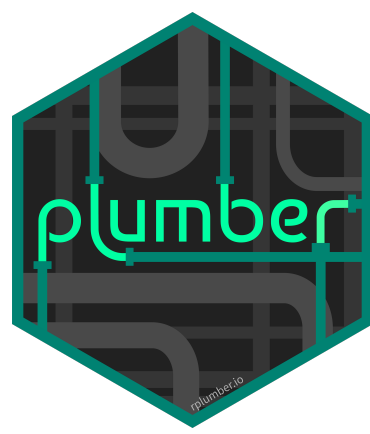


```
future::plan("multisession")
```

```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

```
## @get /slow/<id>  
function(id) {  
  future::future({  
    slow_calc(id)  
  })  
}
```





Receive

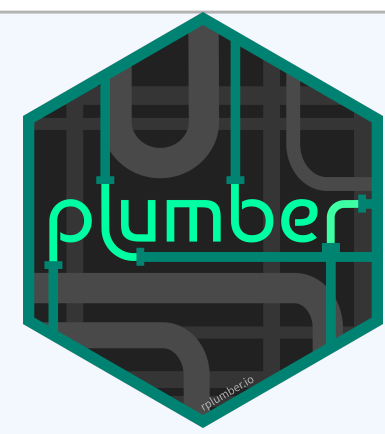
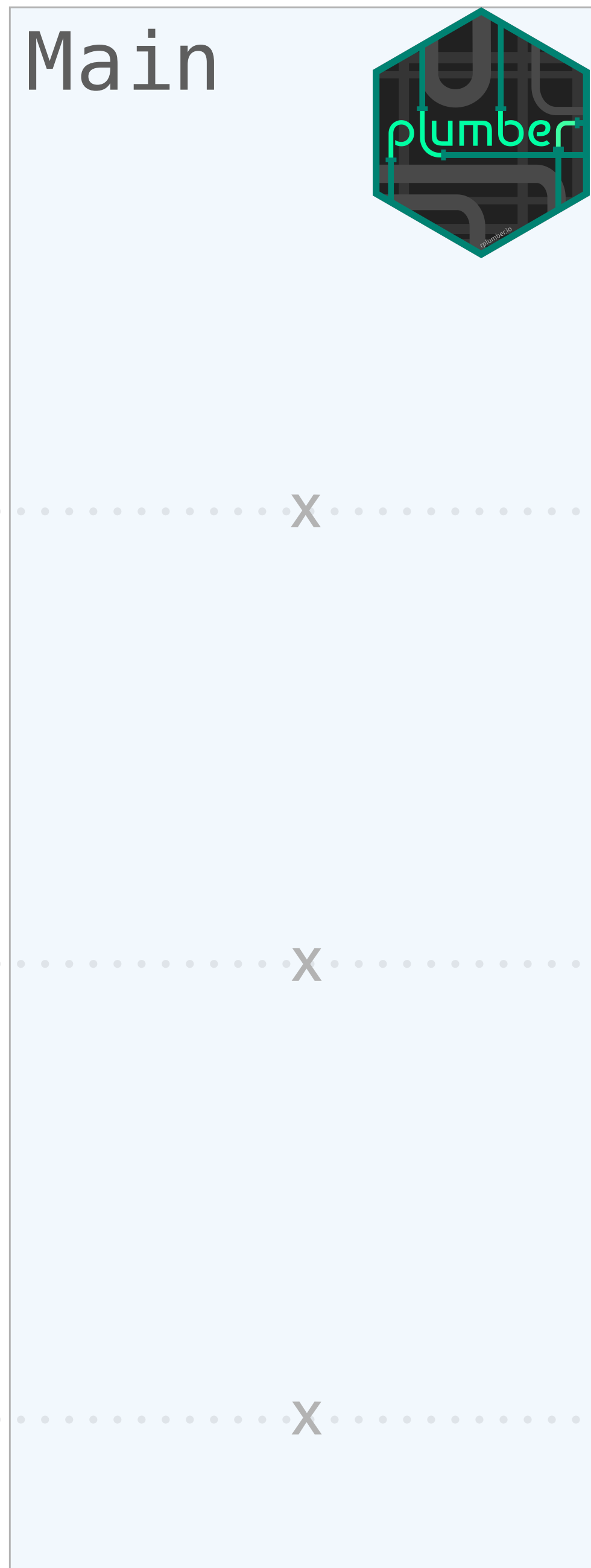


Process



Respond





Receive



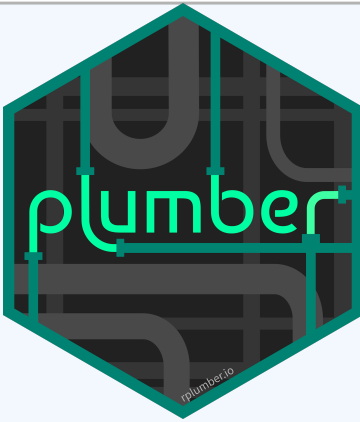
Process



Respond



Main



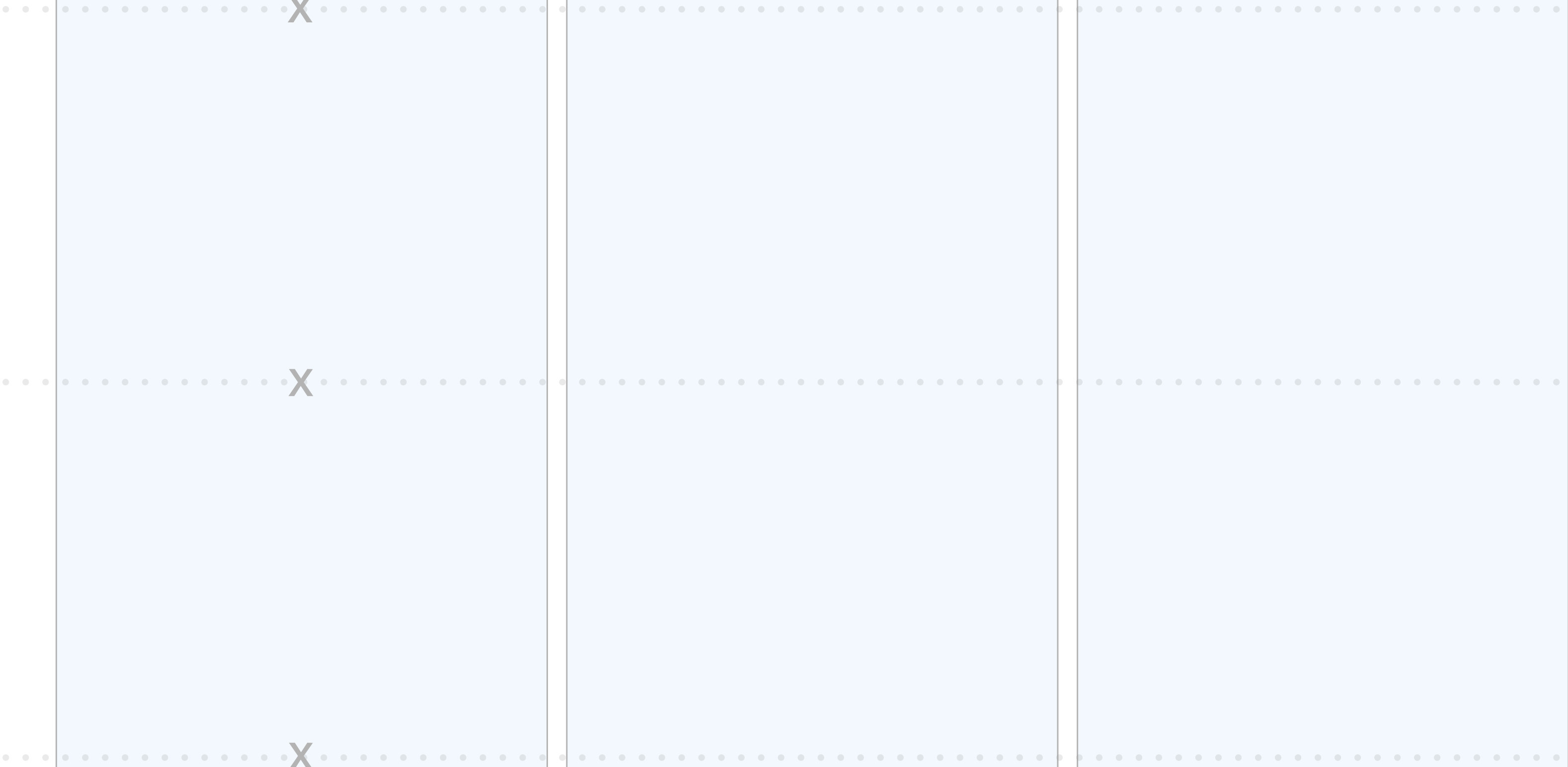
Receive

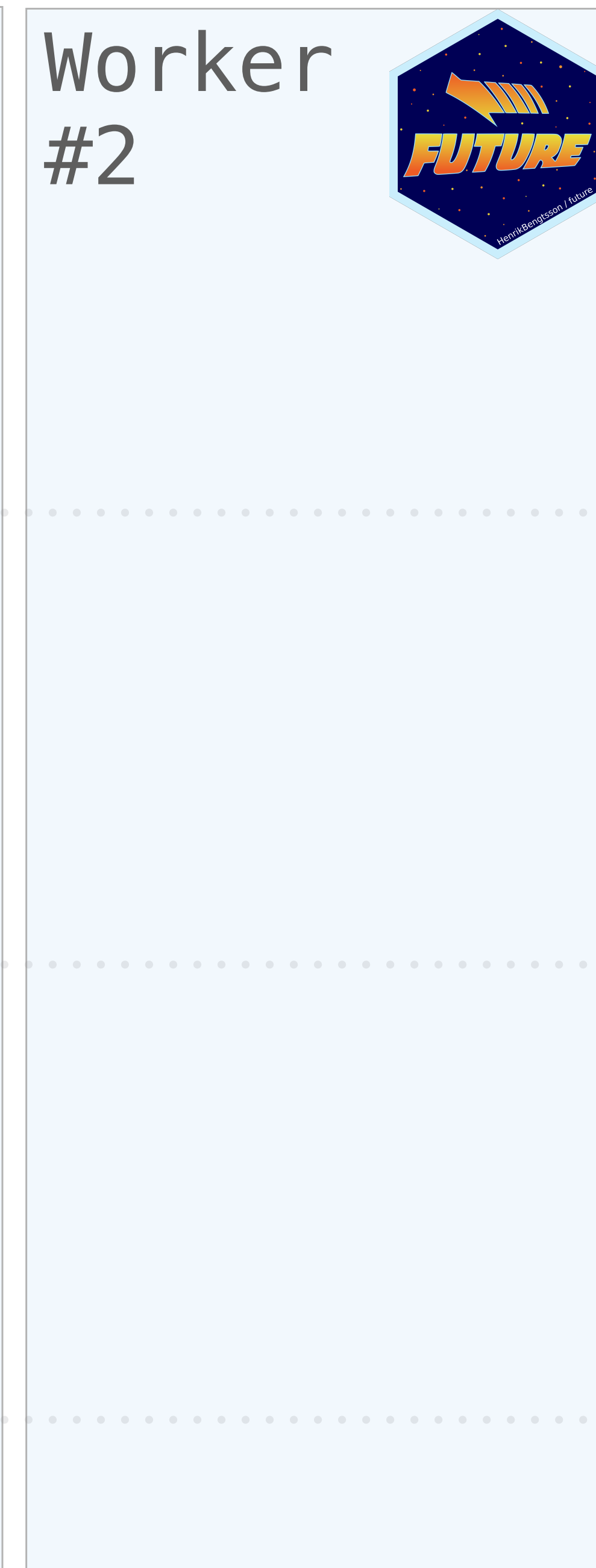
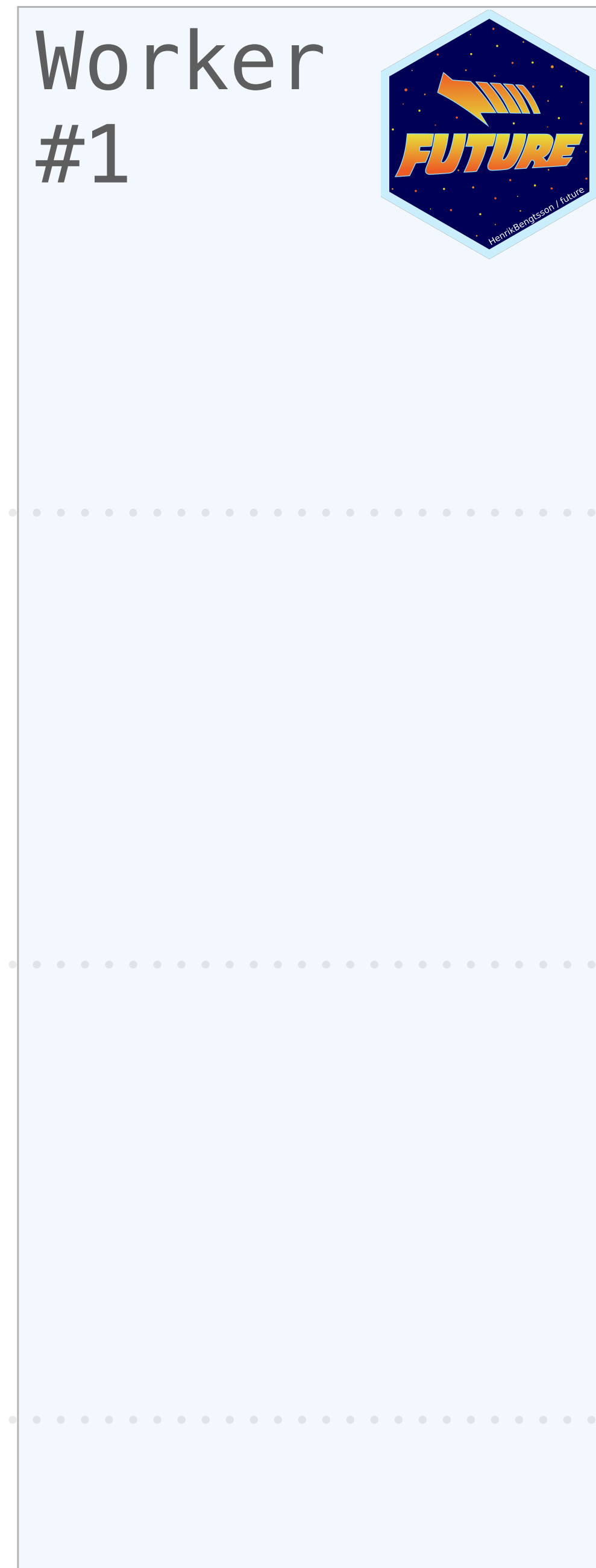


Process



Respond







Main



Worker
#1



Worker
#2



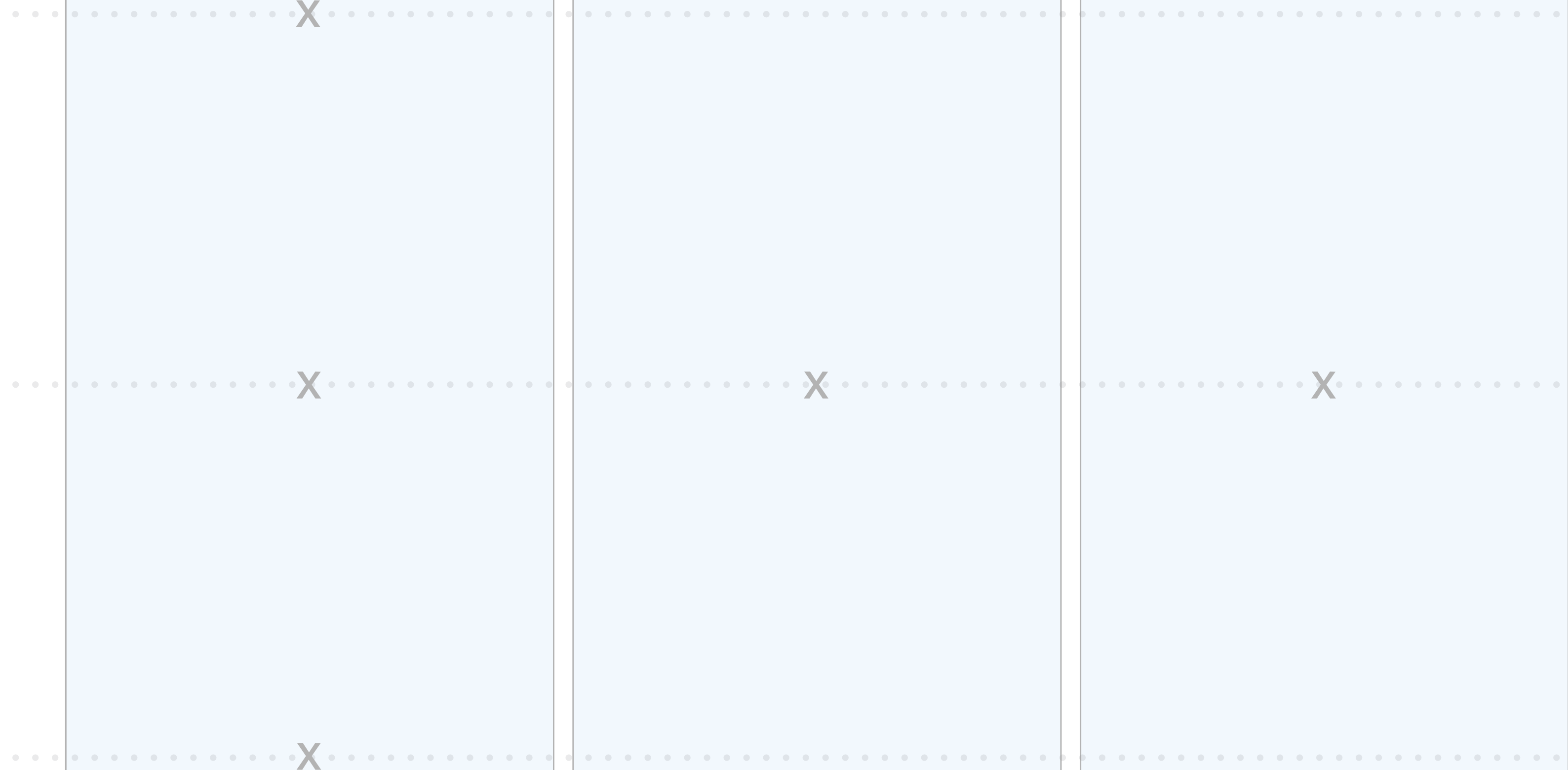
Receive



Process



Respond



Receive



Process



Respond

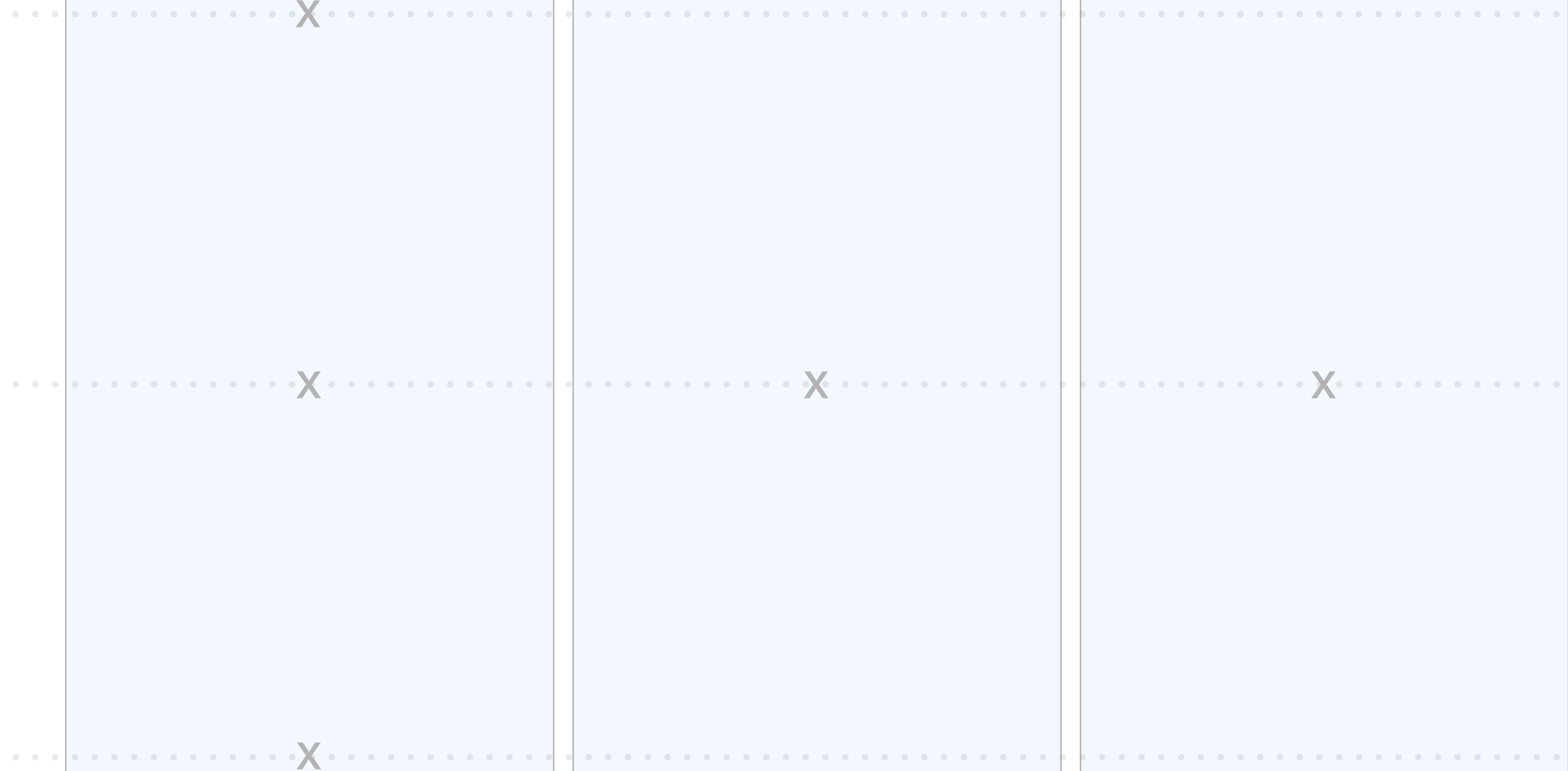
Main



Worker
#1



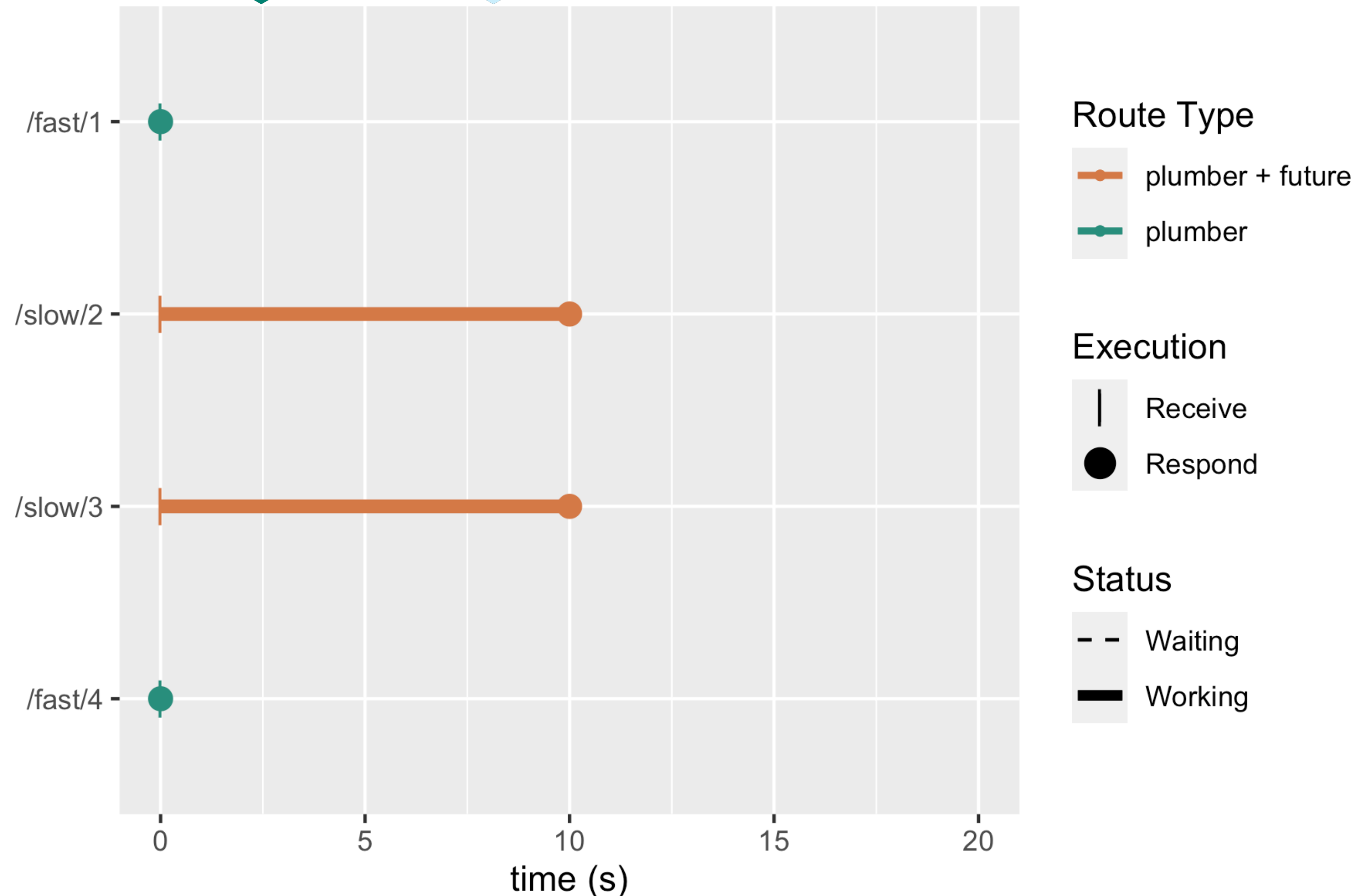
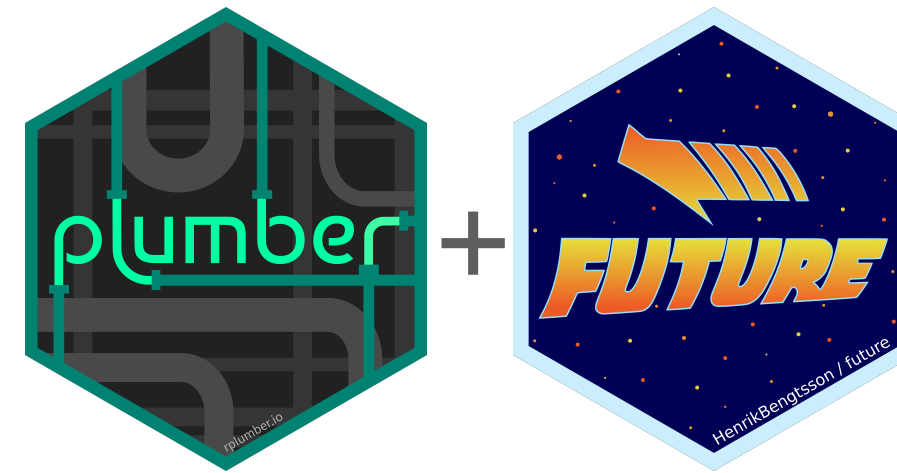
Worker
#2



```
future::plan("multisession")
```

```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

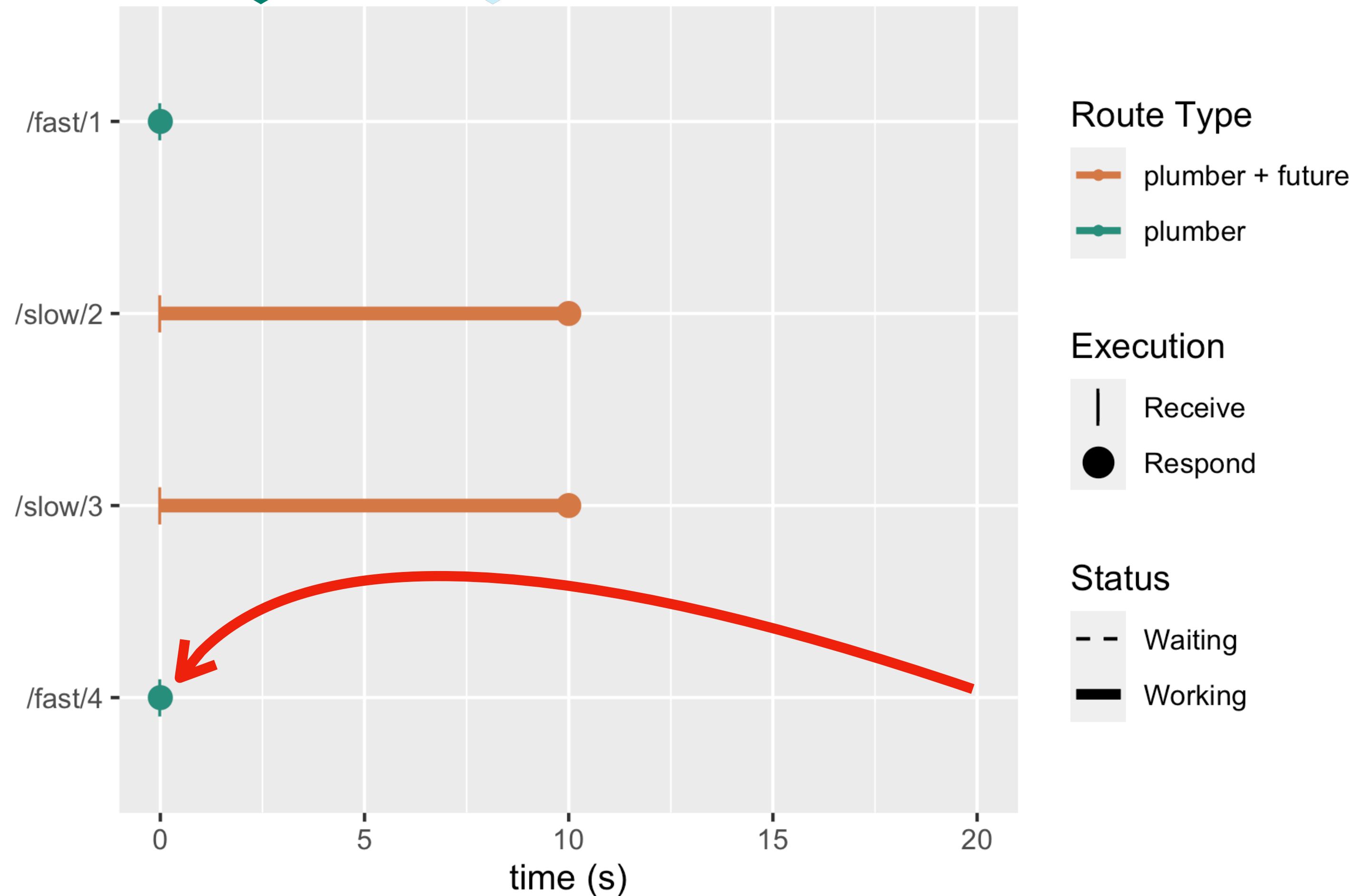
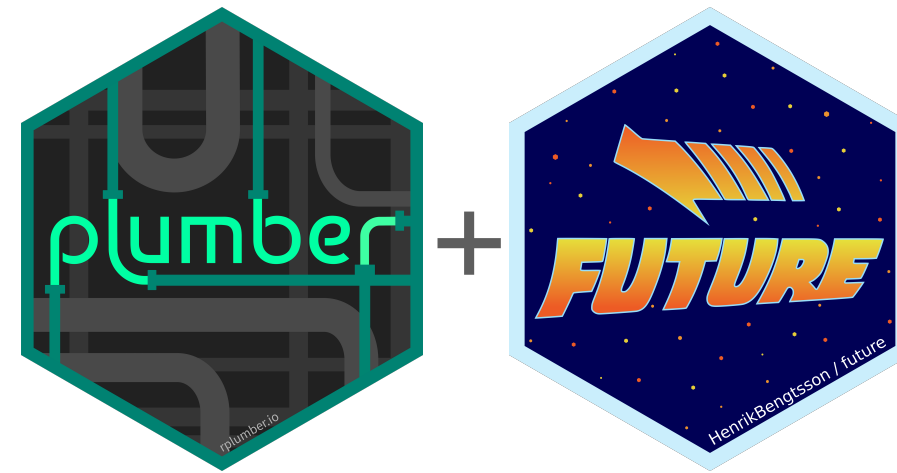
```
## @get /slow/<id>  
function(id) {  
  future::future({  
    slow_calc(id)  
  })  
}
```



```
future::plan("multisession")
```

```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

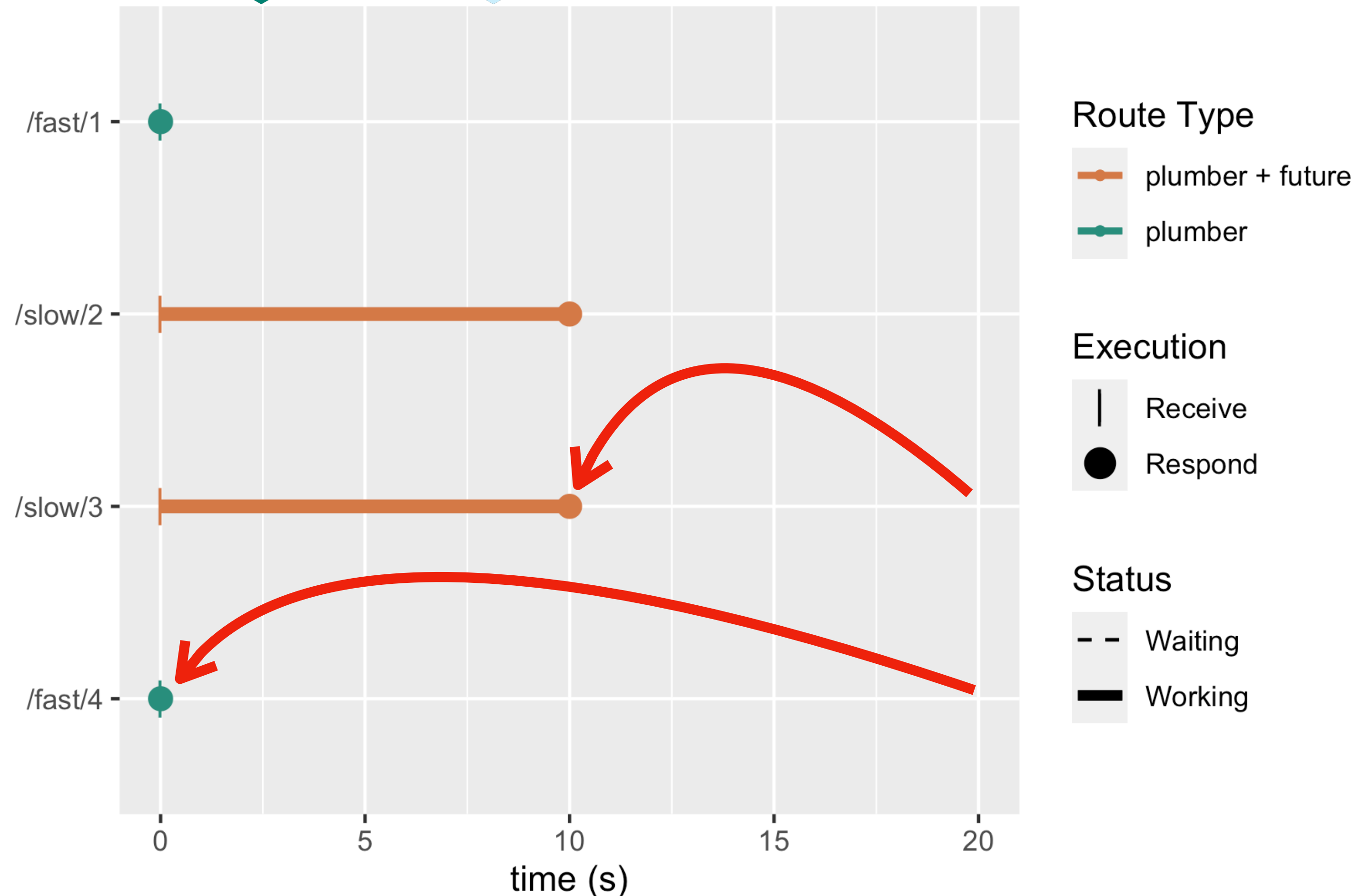
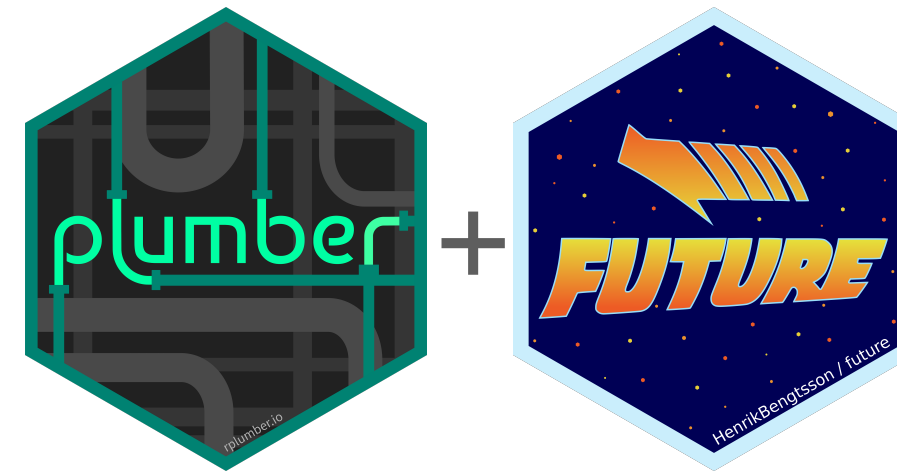
```
## @get /slow/<id>  
function(id) {  
  future::future({  
    slow_calc(id)  
  })  
}
```



```
future::plan("multisession")
```

```
## @get /fast/<id>  
function(id) {  
  fast_calc(id)  
}
```

```
## @get /slow/<id>  
function(id) {  
  future::future({  
    slow_calc(id)  
  })  
}
```



Limitations...

Limitations...

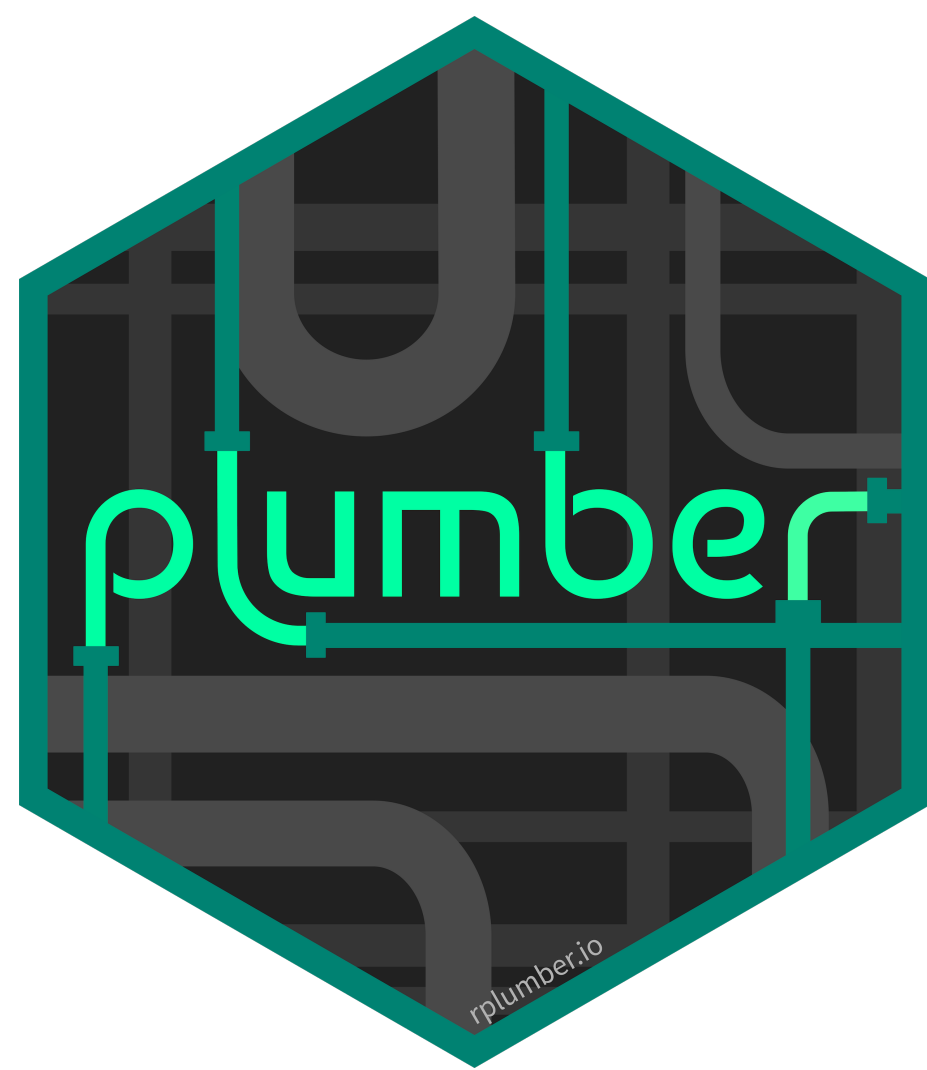
- Manually add `future(expr)`

Limitations...

- Manually add `future(expr)`
- `future(expr)` is not instant



Limitations...

- Manually add `future(expr)`
- `future(expr)` is not instant
- Processing power is finite
Memory size is finite



+



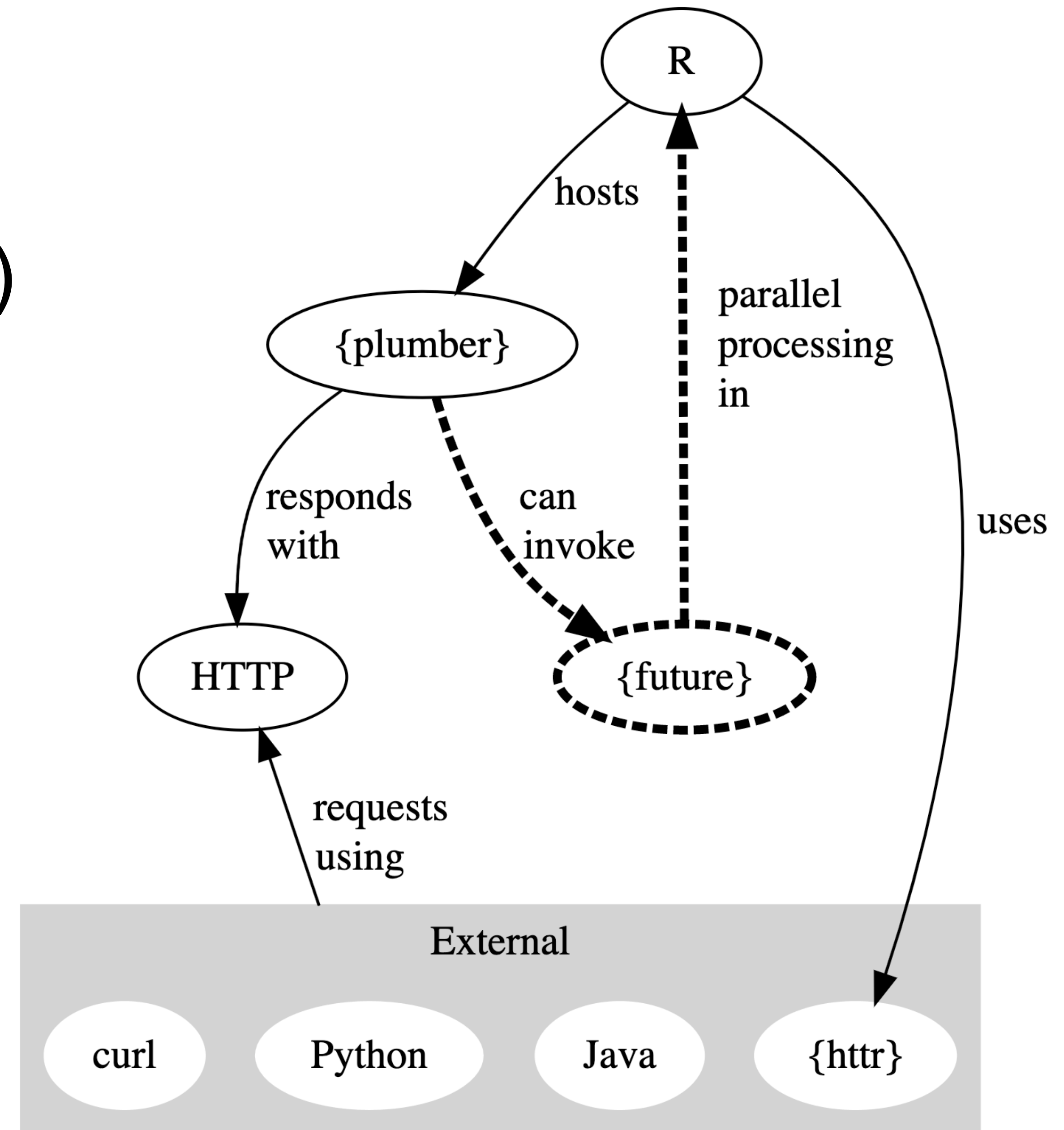
Barret Schloerke
RStudio / Shiny Team
  @schloerke

- Offload *slow* routes using `future(expr)`
- **Slides and material:**
rstudio.io/global2021/barretschloerke

```

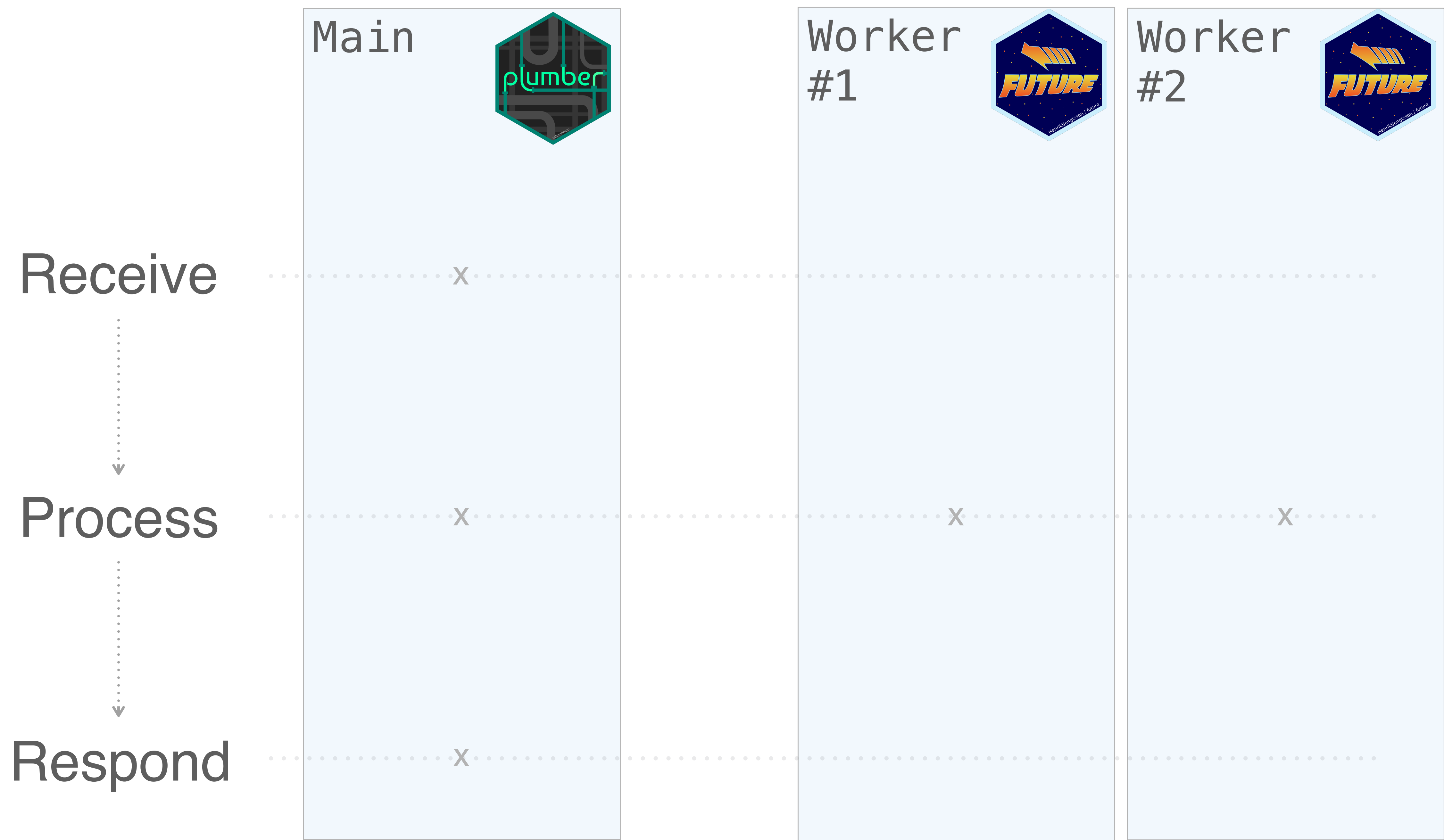
#* @get /ask
function(question) {
  if (is_quick(question))
    barret::answer(
      question)
  else
    future({
      barret::answer(
        question)
    })
}

```

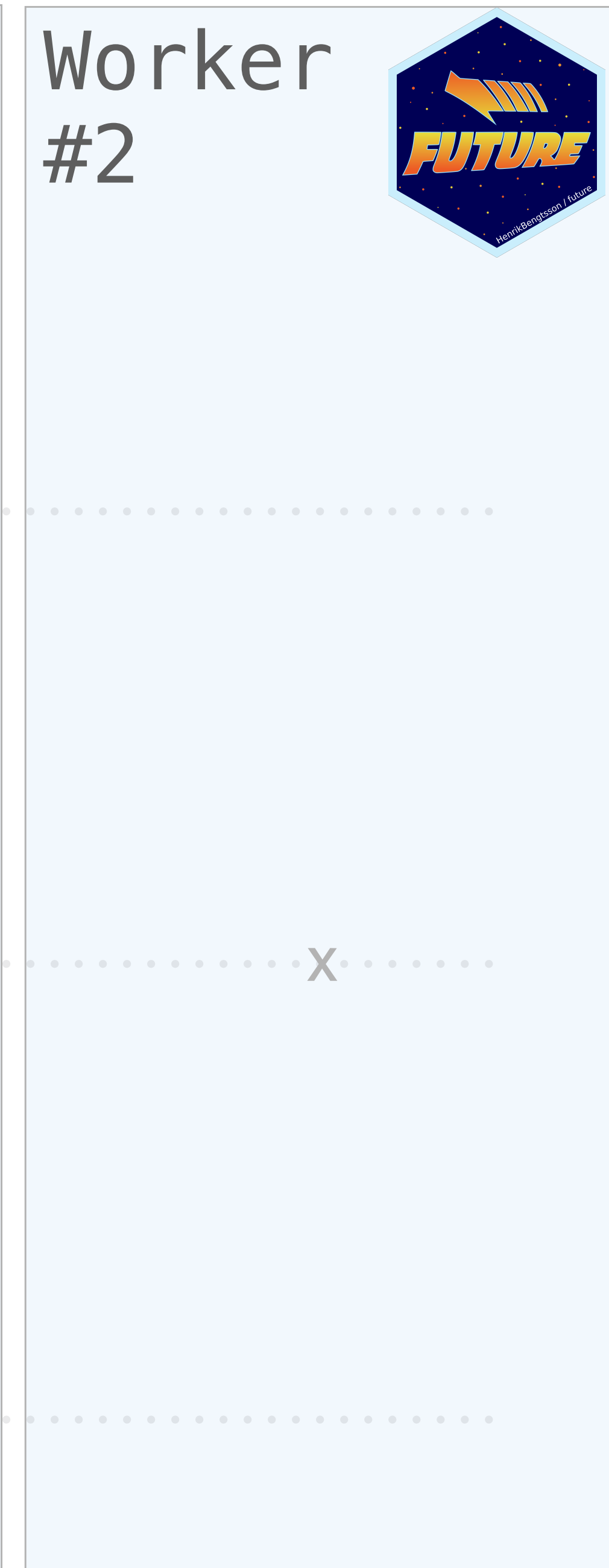
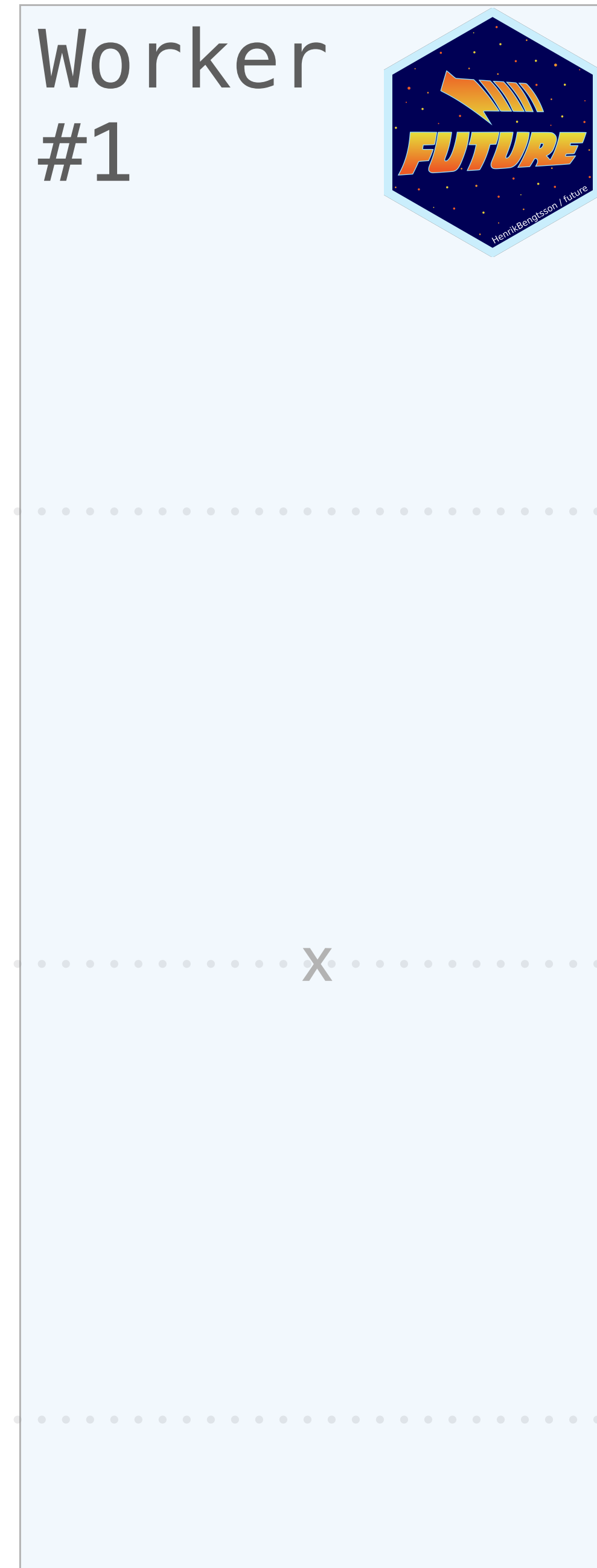
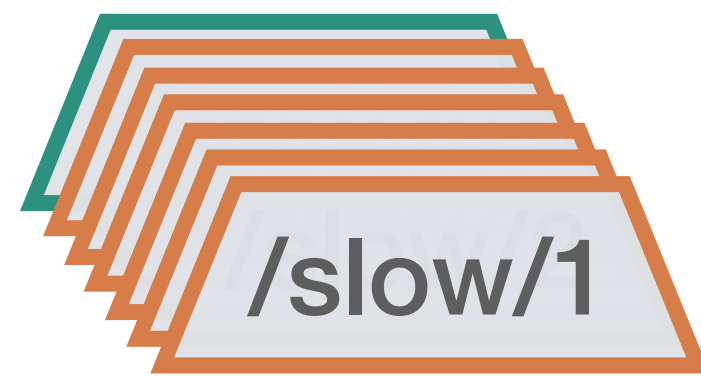


Demo:

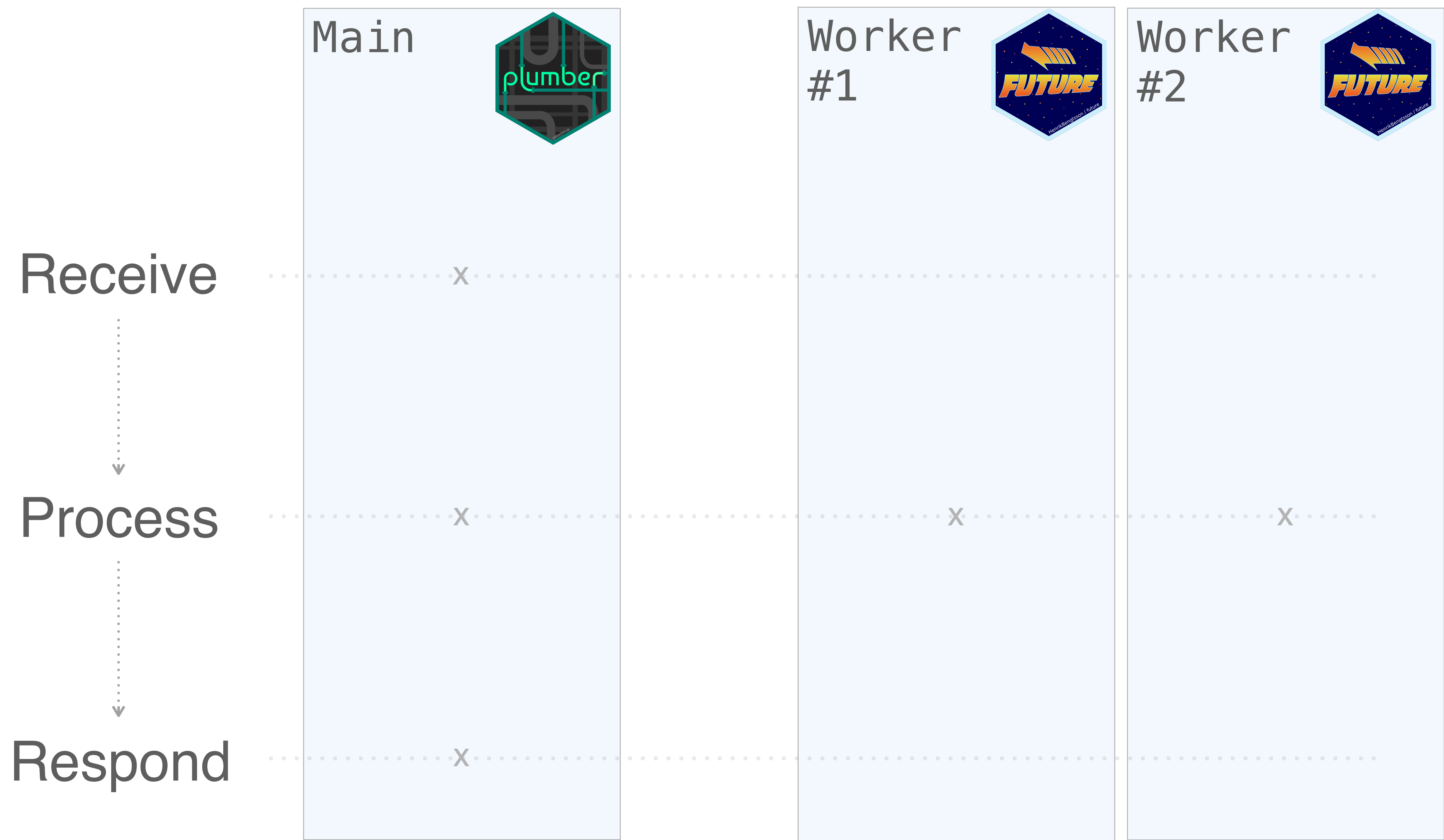
What happens when I run out of workers?



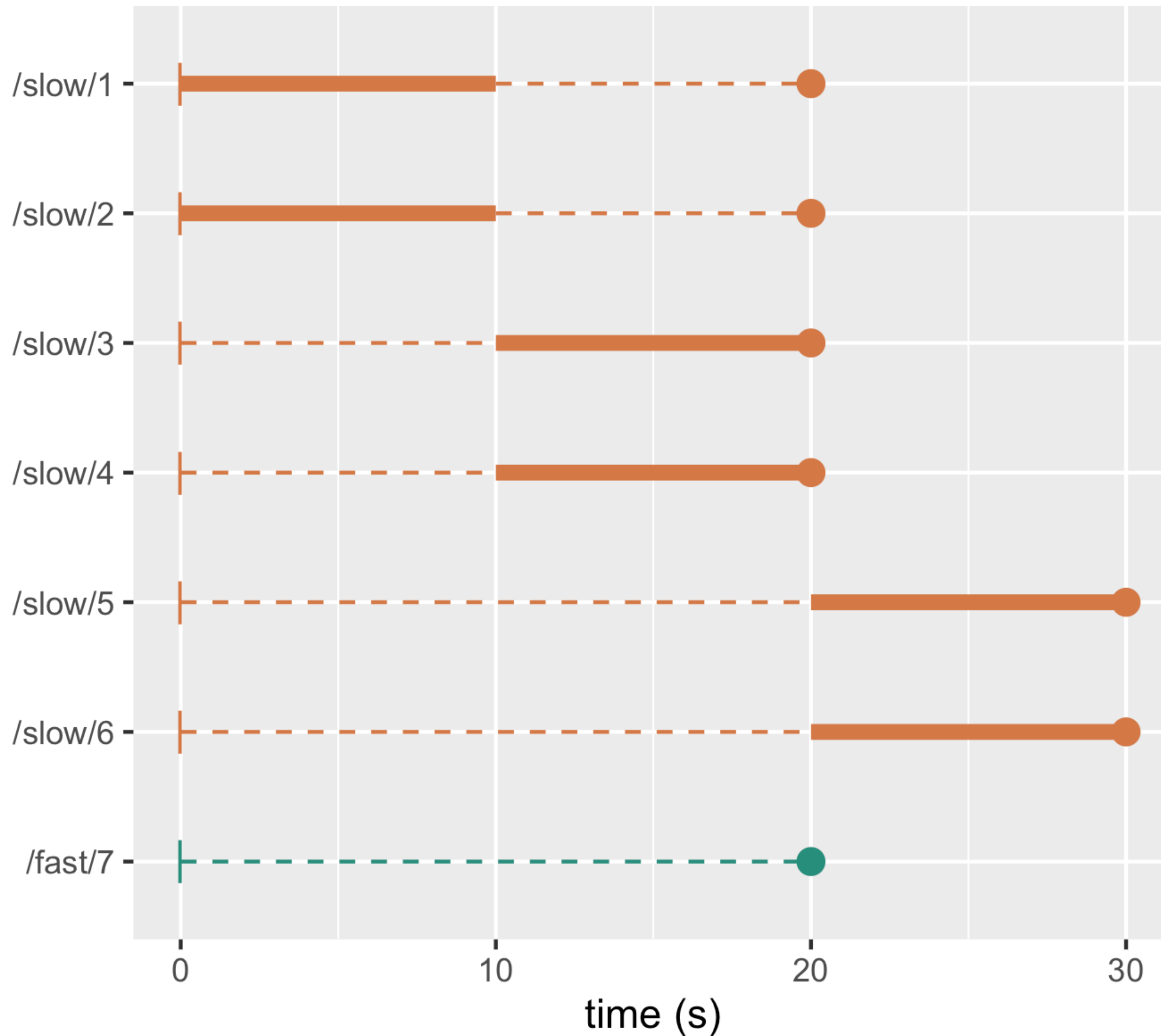
workers = 2



workers = 2



workers = 2



Route Type

- plumber + future
- plumber

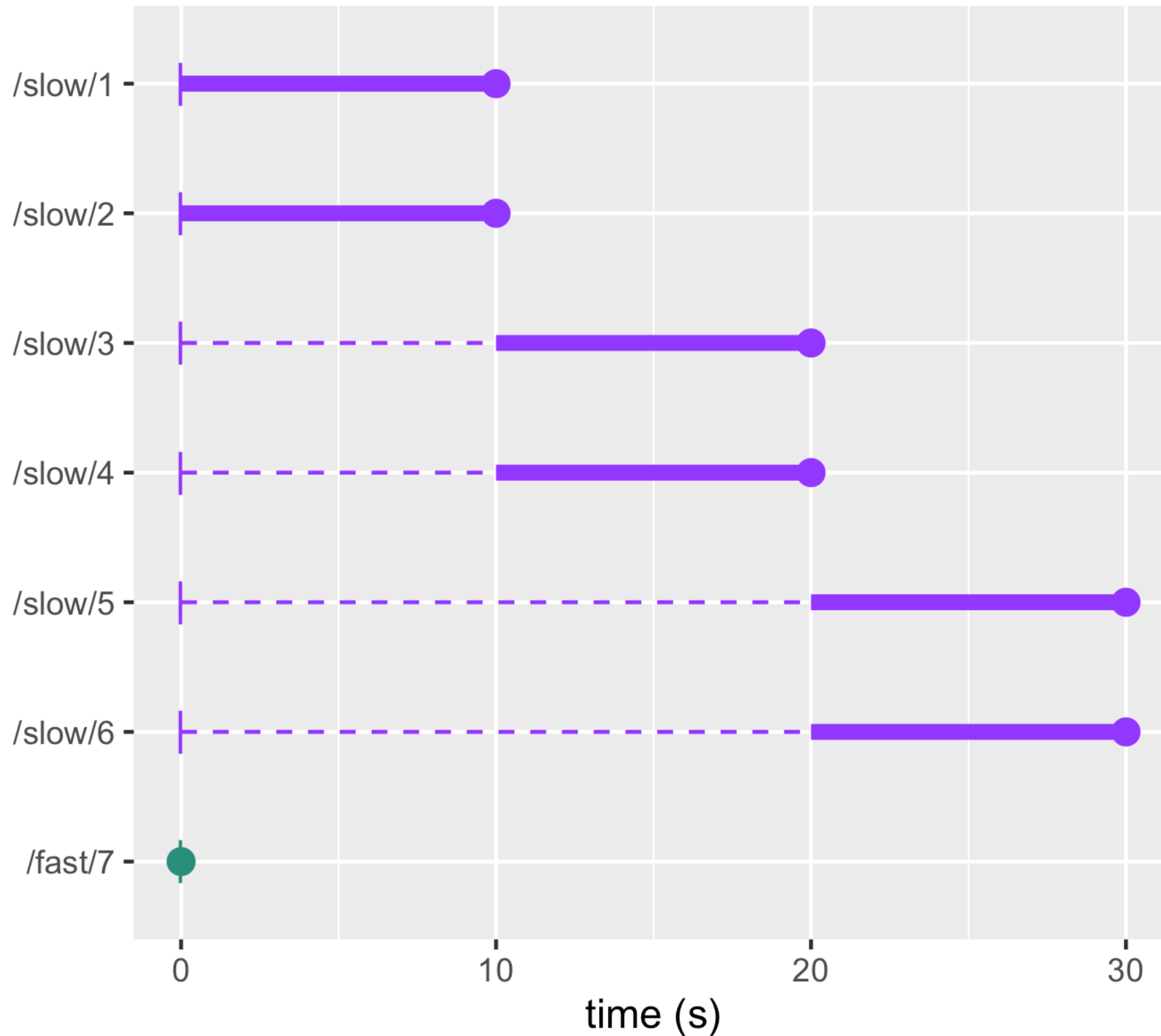
Execution

- Receive
- Respond

Status

- Waiting
- Working

workers = 2



Route Type

- plumber + future_promise
- plumber

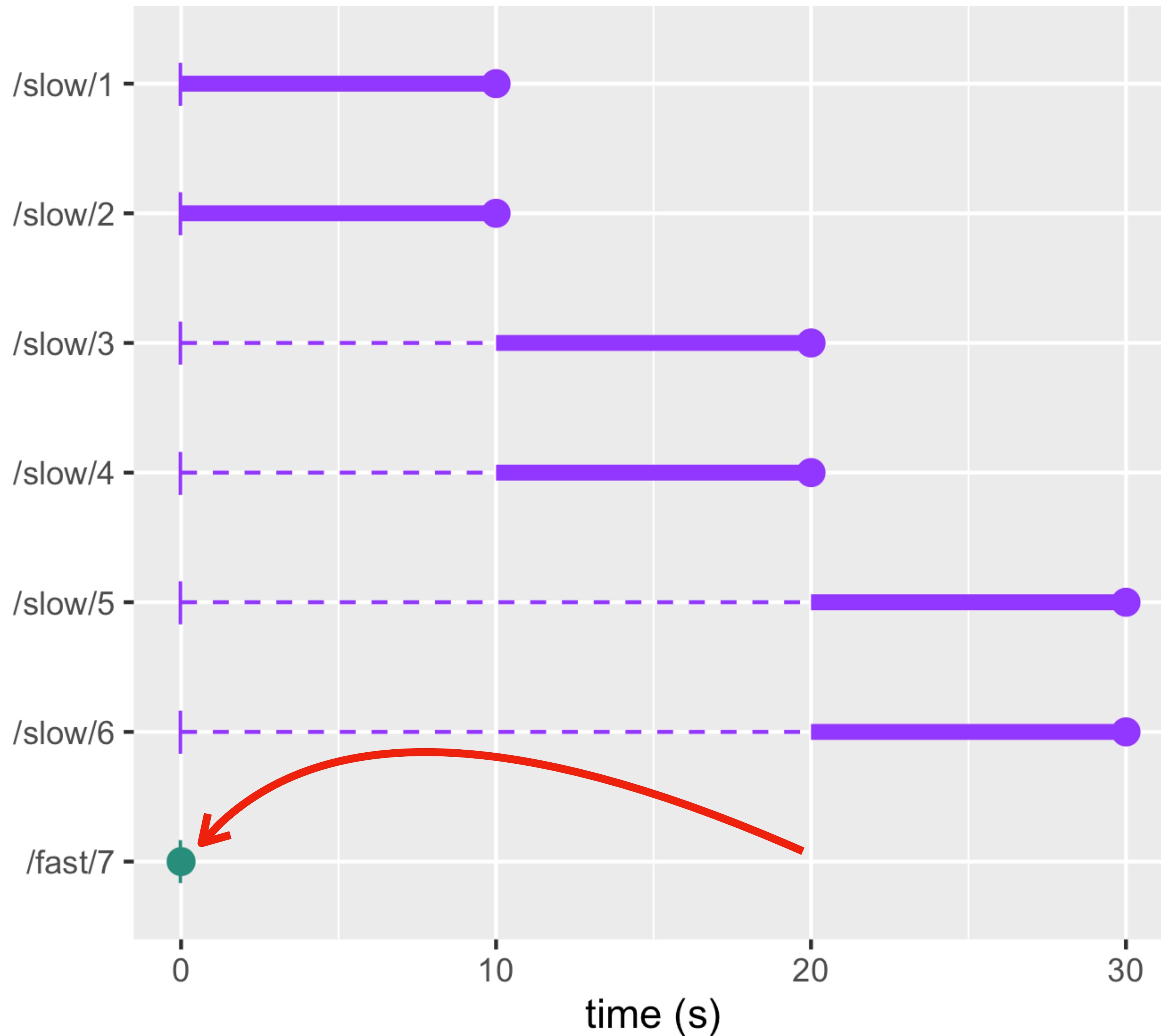
Execution

- Receive
- Respond

Status

- Waiting in promise
- Working in future

workers = 2



Route Type

- plumber + future_promise
- plumber

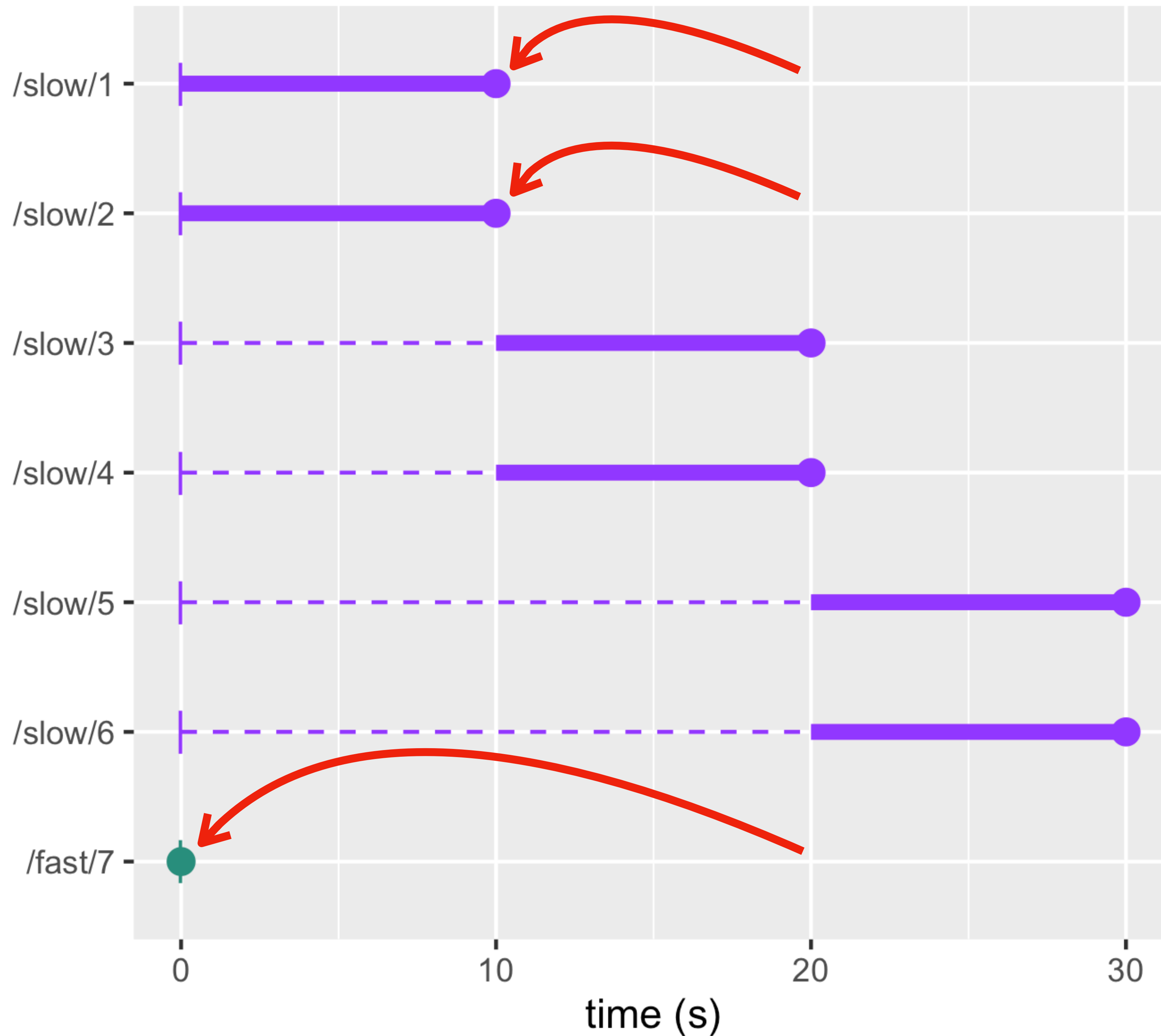
Execution

- Receive
- Respond

Status

- Waiting in promise
- Working in future

workers = 2



Route Type

- plumber + future_promise
- plumber

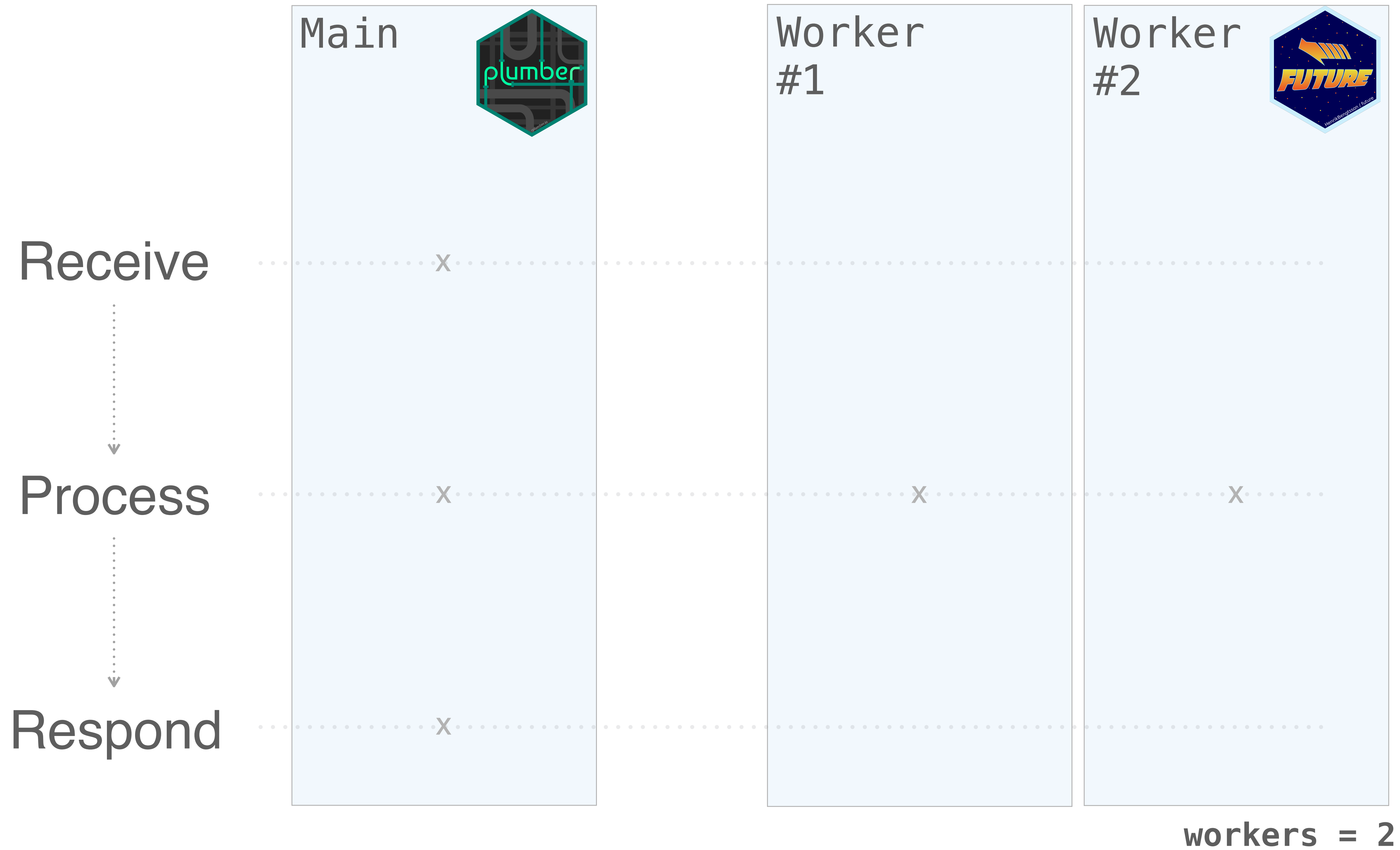
Execution

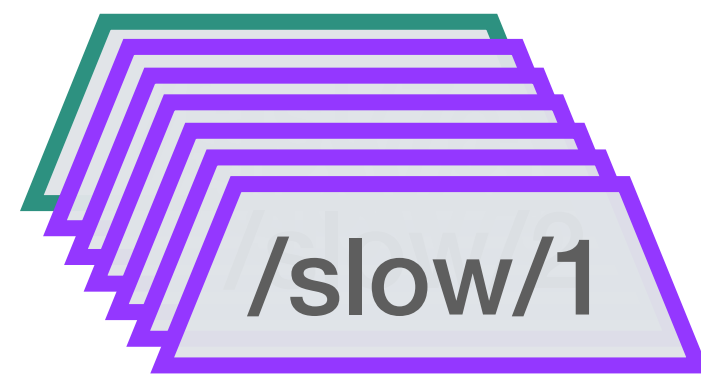
- Receive
- Respond

Status

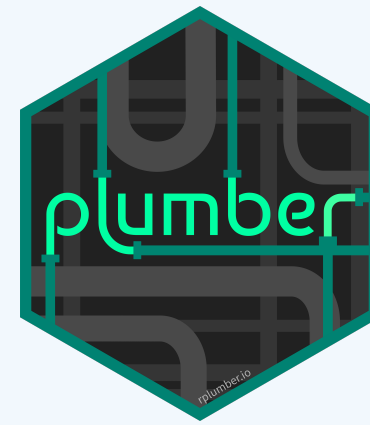
- Waiting in promise
- Working in future

workers = 2





Main



Worker
#1

Worker
#2



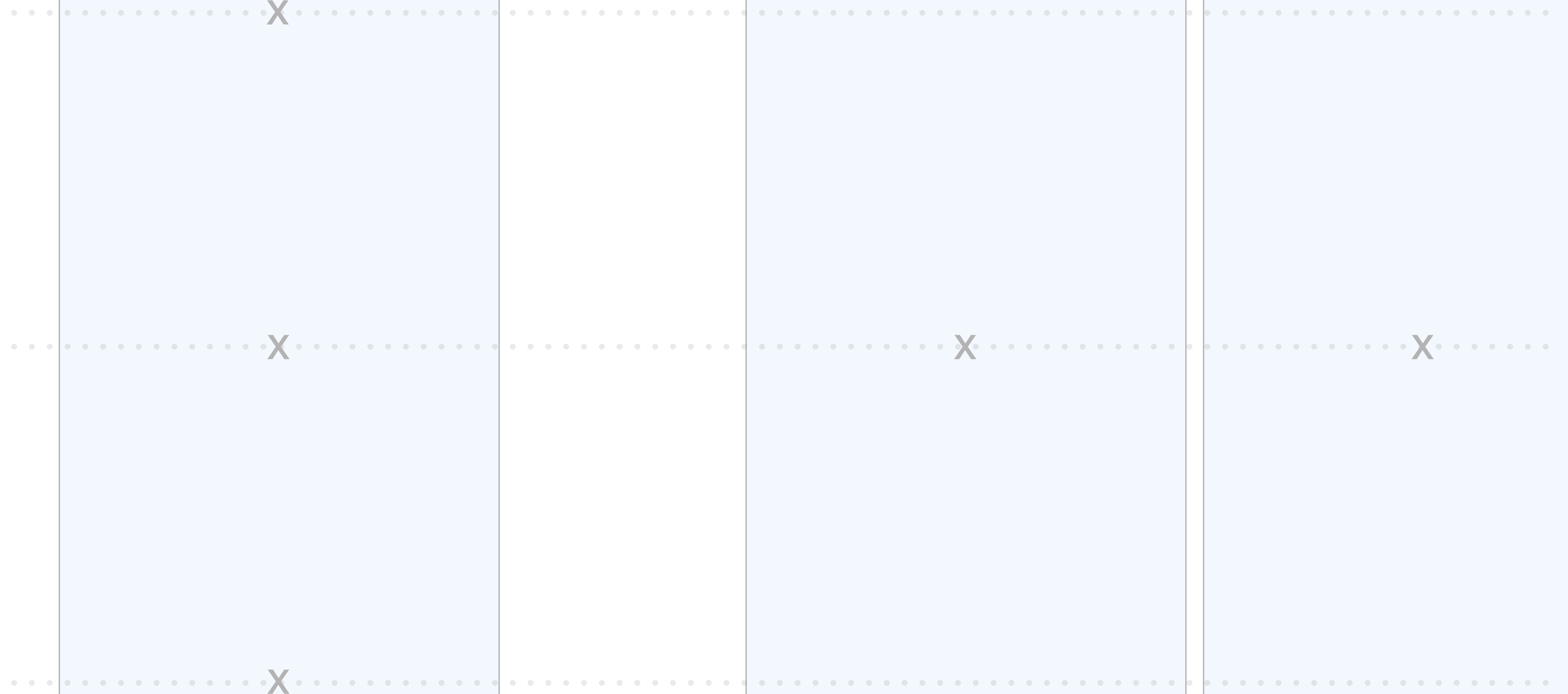
Receive



Process



Respond



workers = 2

