

# Understanding Garbage Collectors in object-oriented programming

Lecture 1 - Introduction

Nahuel Palumbo

✉ [nahuel.palumbo@inria.fr](mailto:nahuel.palumbo@inria.fr)



# **Memory Management**

## **Garbage Collection**



# Manually Memory Management

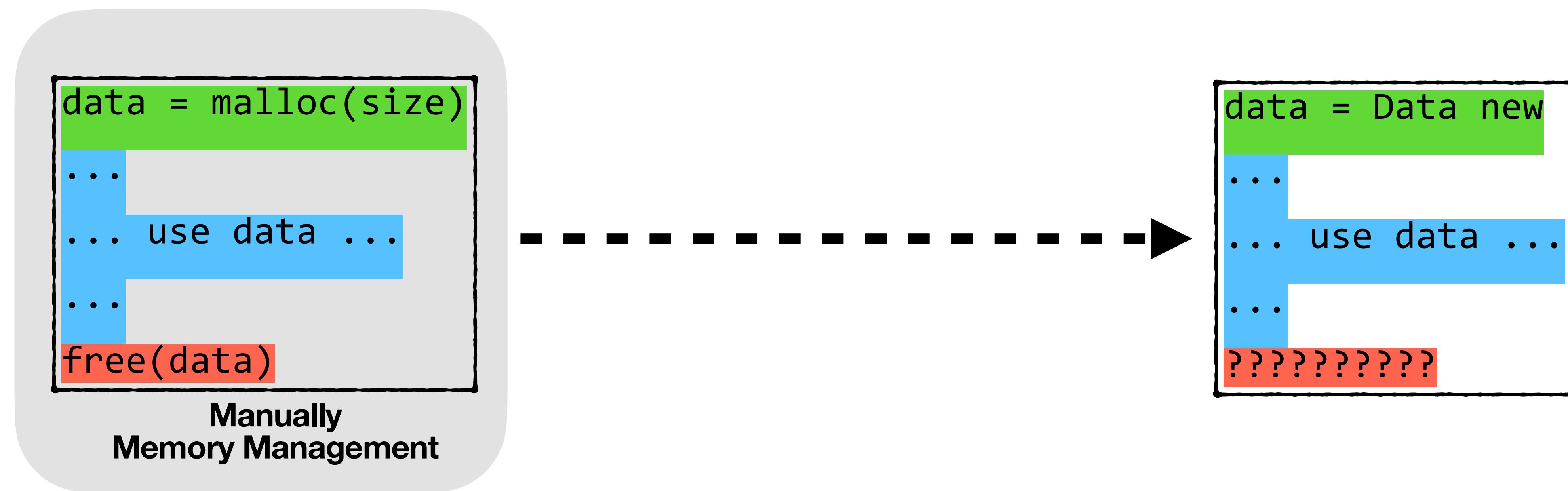
## A work for devs?

```
data = malloc(size)  
...  
... use data ...  
...  
free(data)
```

**Manually  
Memory Management**



# Automatic Memory Management



# Automatic Memory Management

## Garbage Collectors

```
data = malloc(size)  
...  
... use data ...  
...  
free(data)
```

Manually  
Memory Management



```
data = Data new  
...  
... use data ...  
...  
????????????
```

Compute the size  
Allocate in the memory



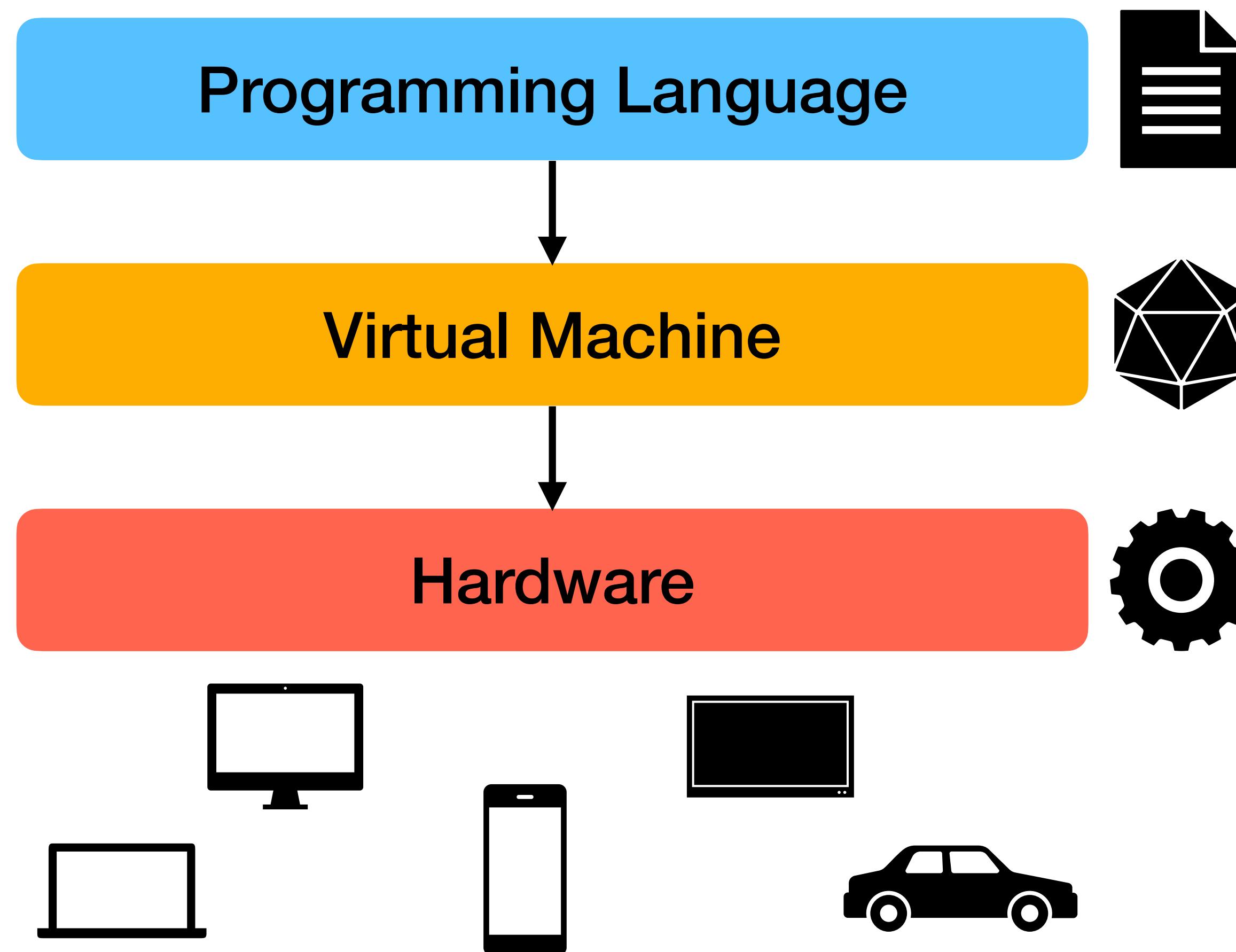
Maybe move the data for  
better use of the memory



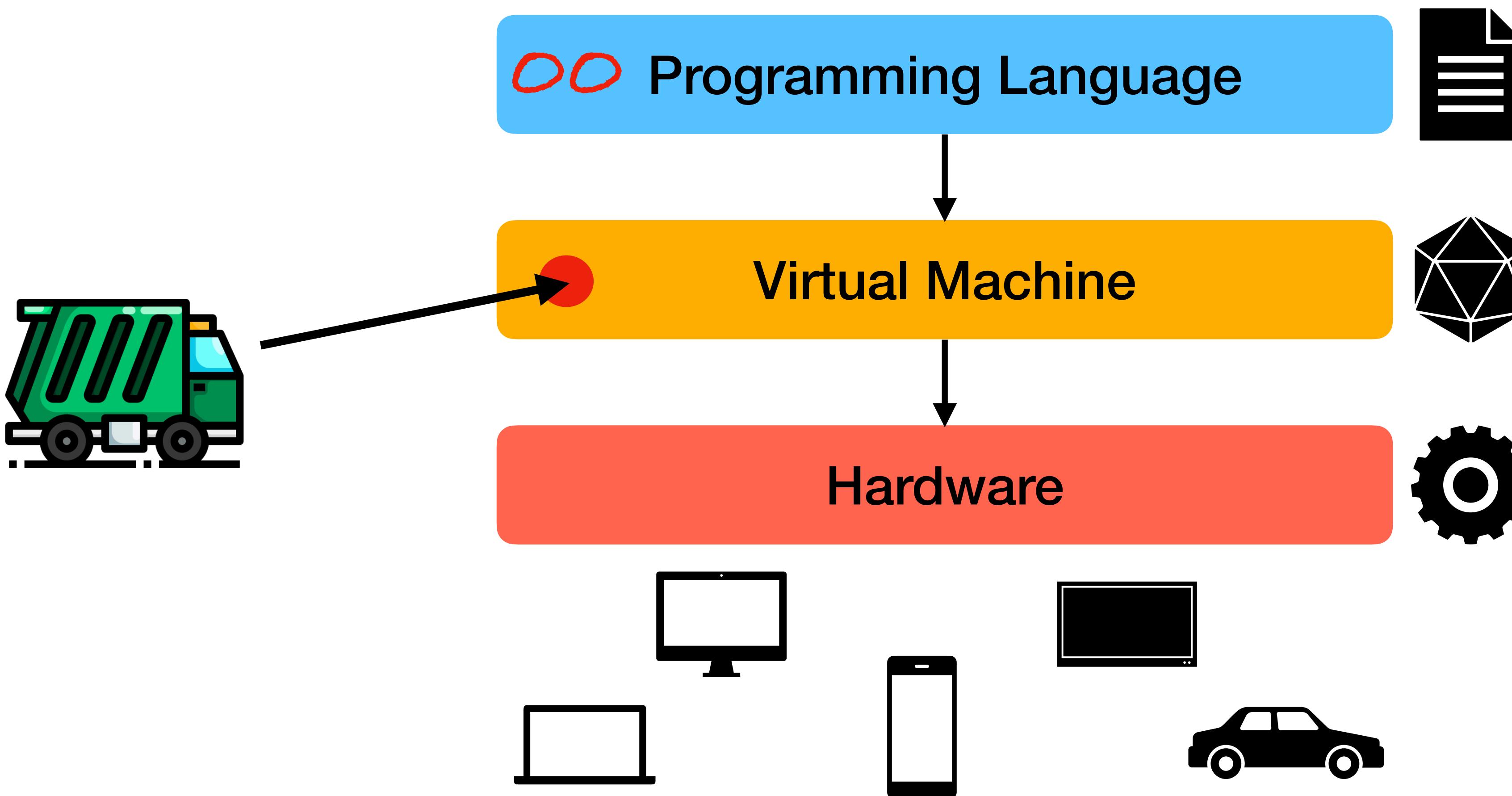
Free the space when data  
is not used anymore



# Language Virtual Machines



# Language Virtual Machines

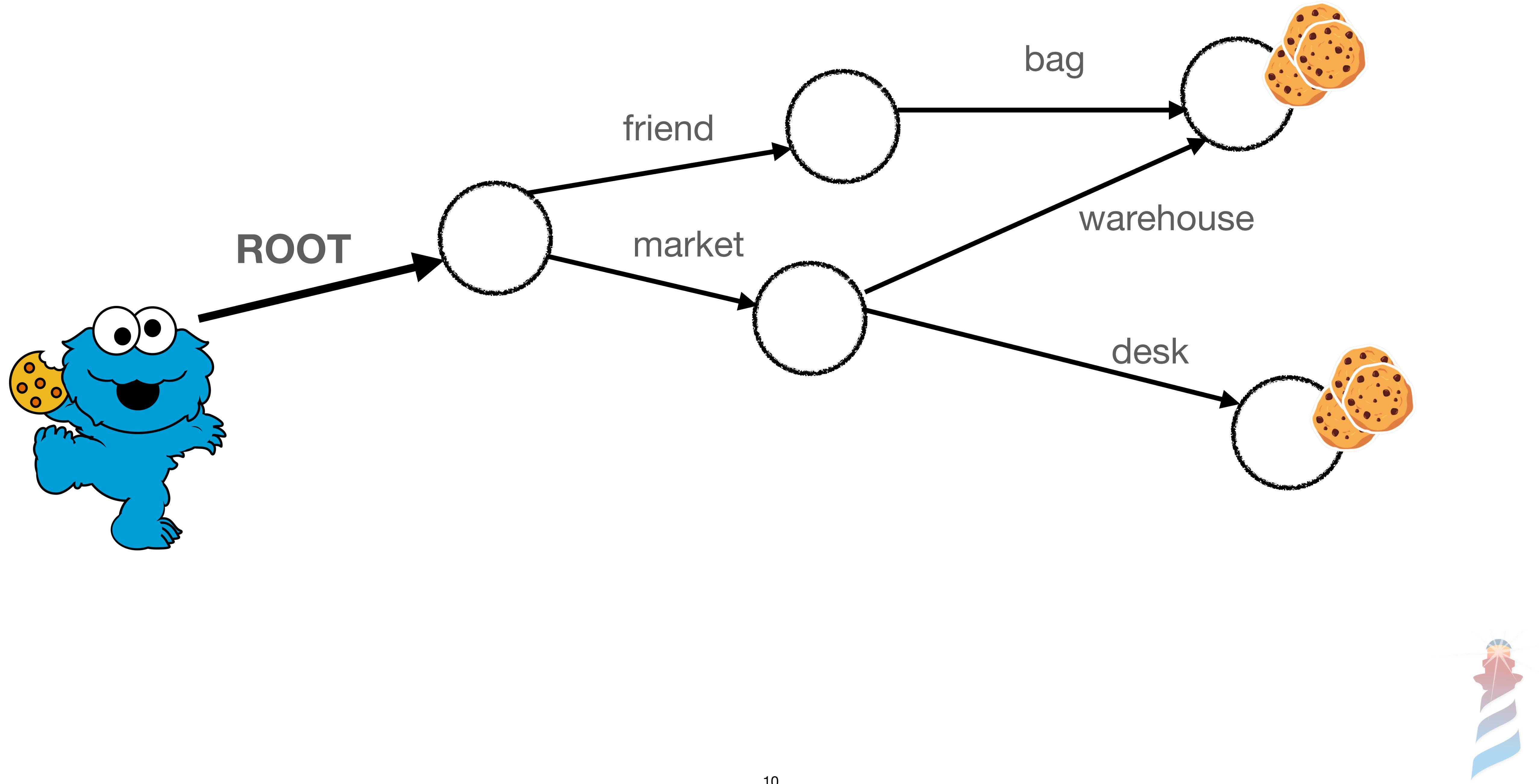


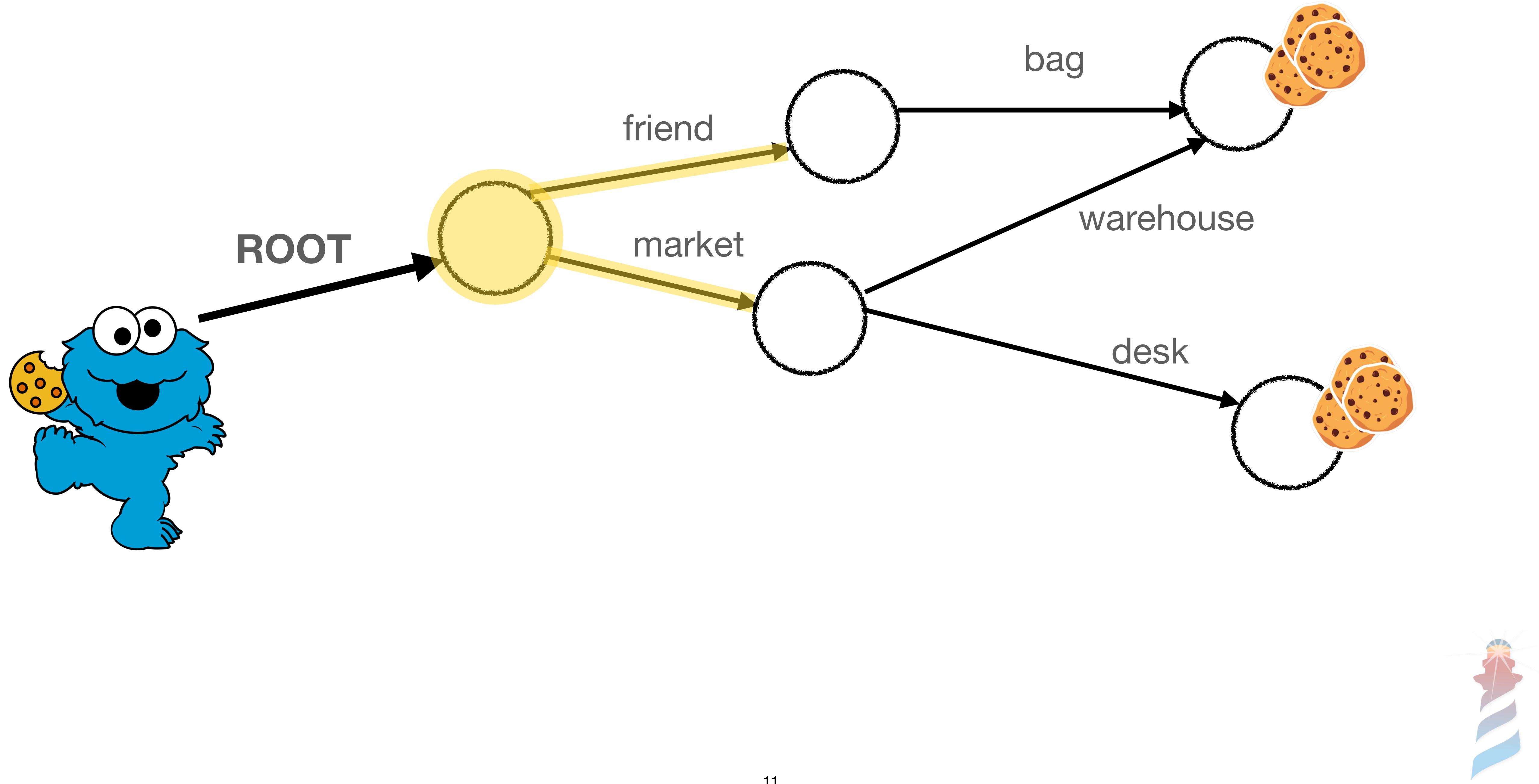
# How Garbage Collectors works in object-oriented programs?

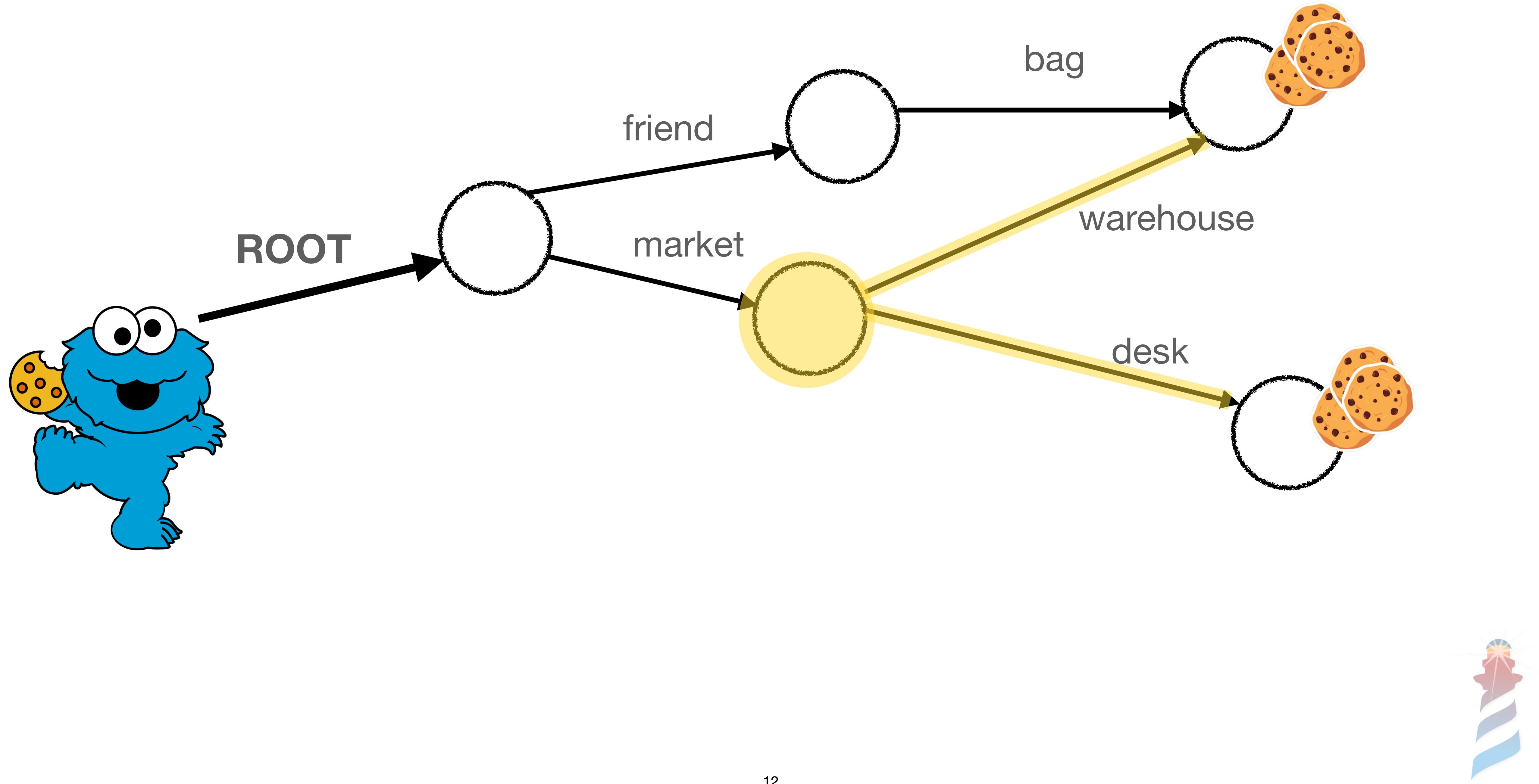


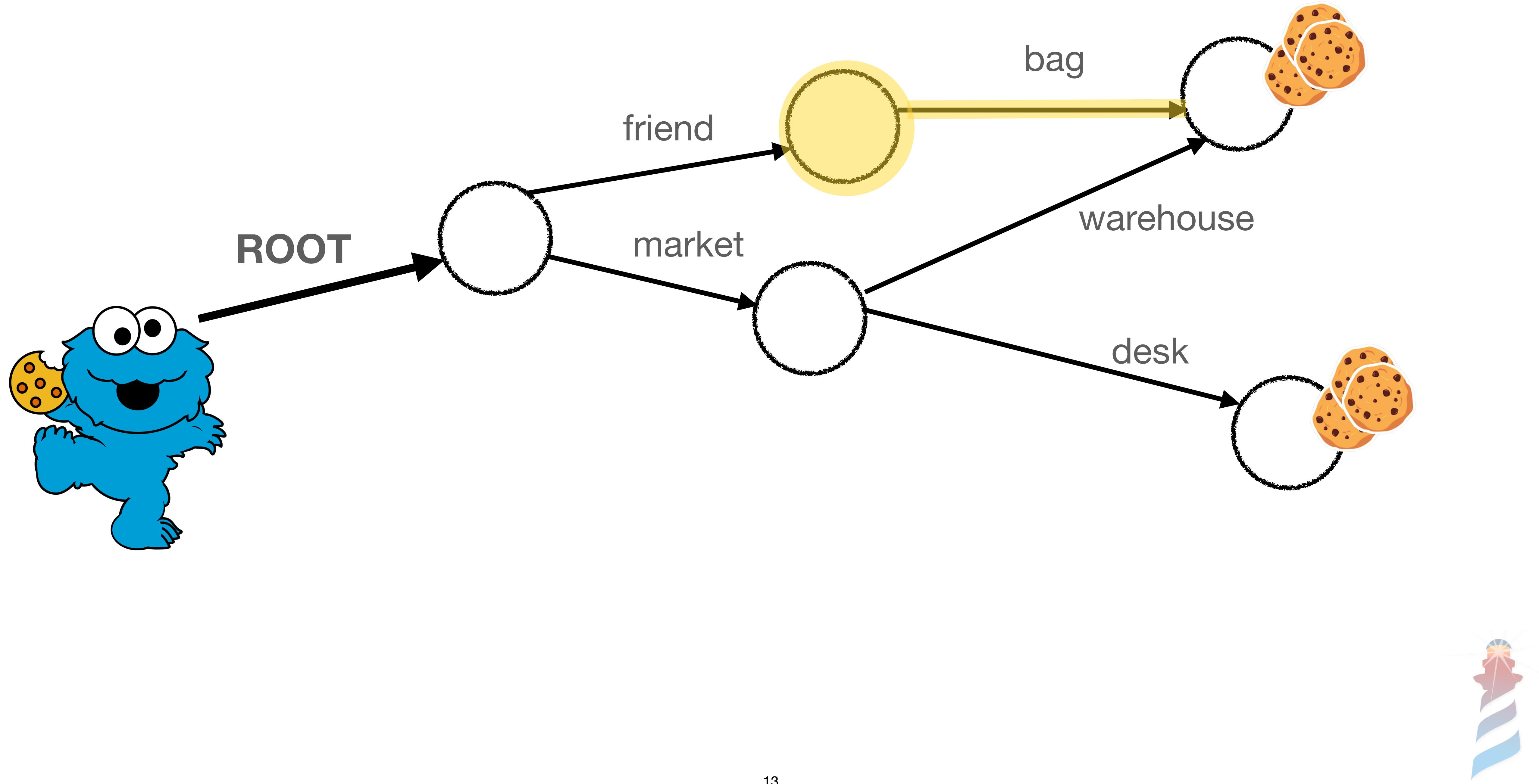
**Let's start with a game...**

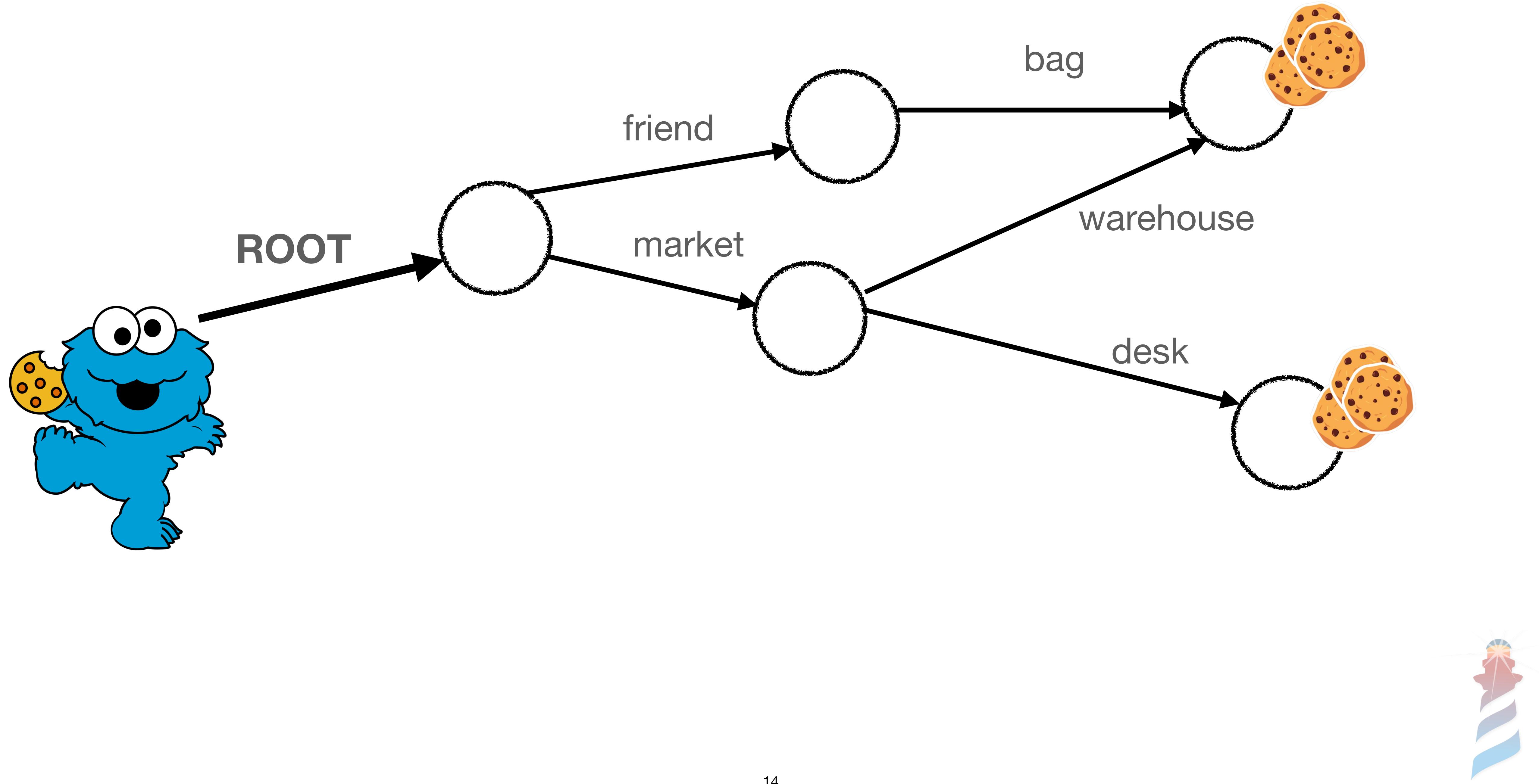


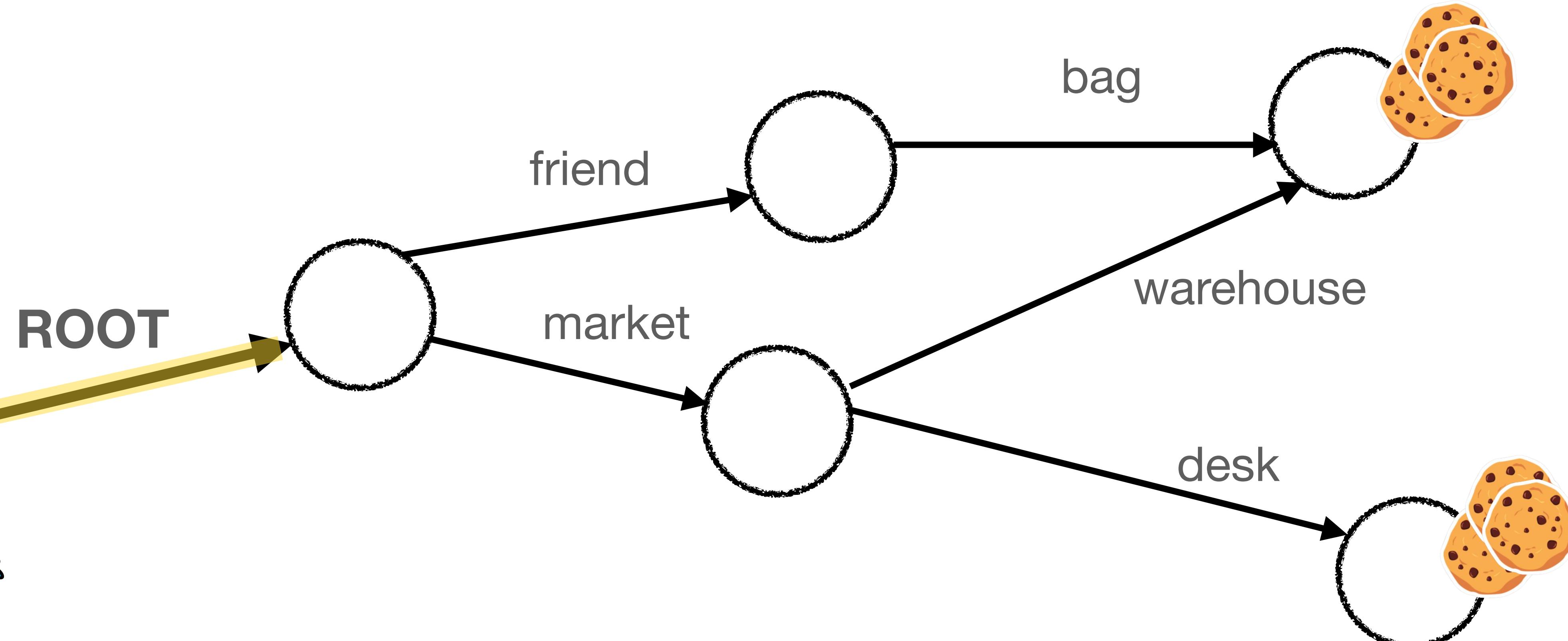


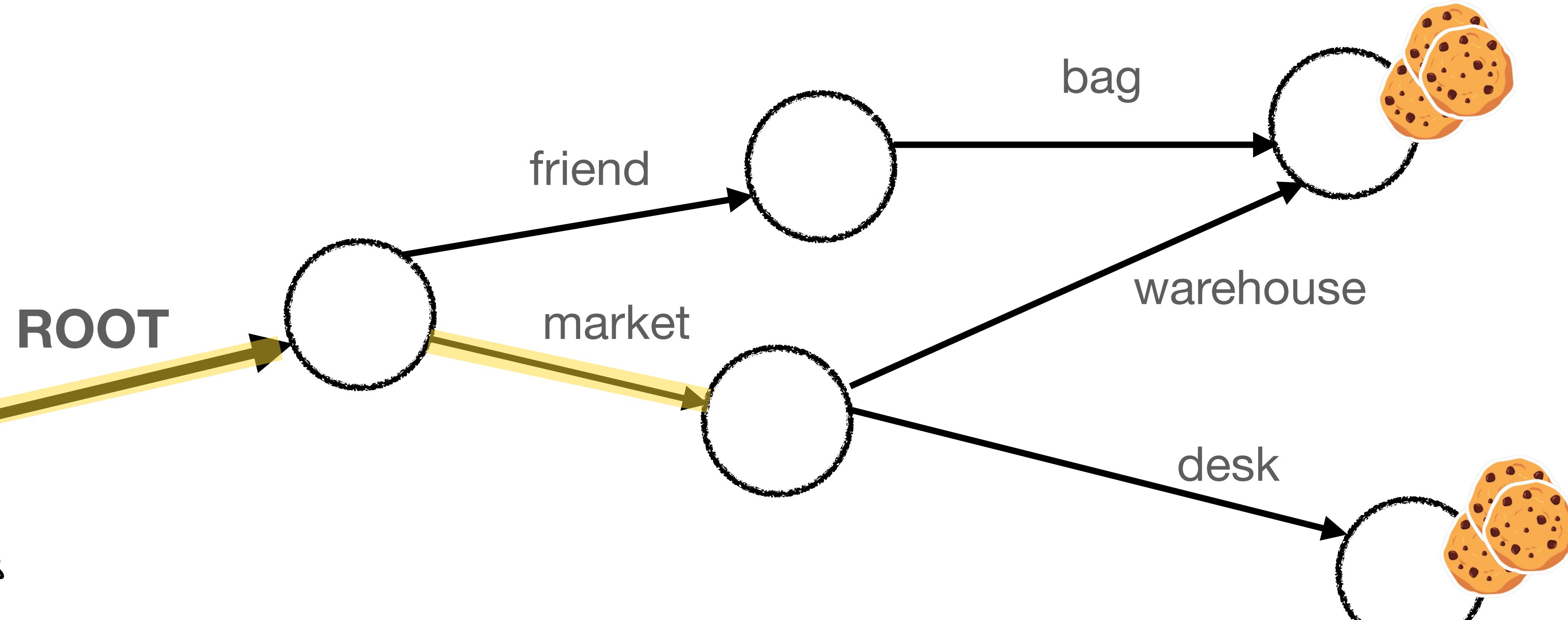


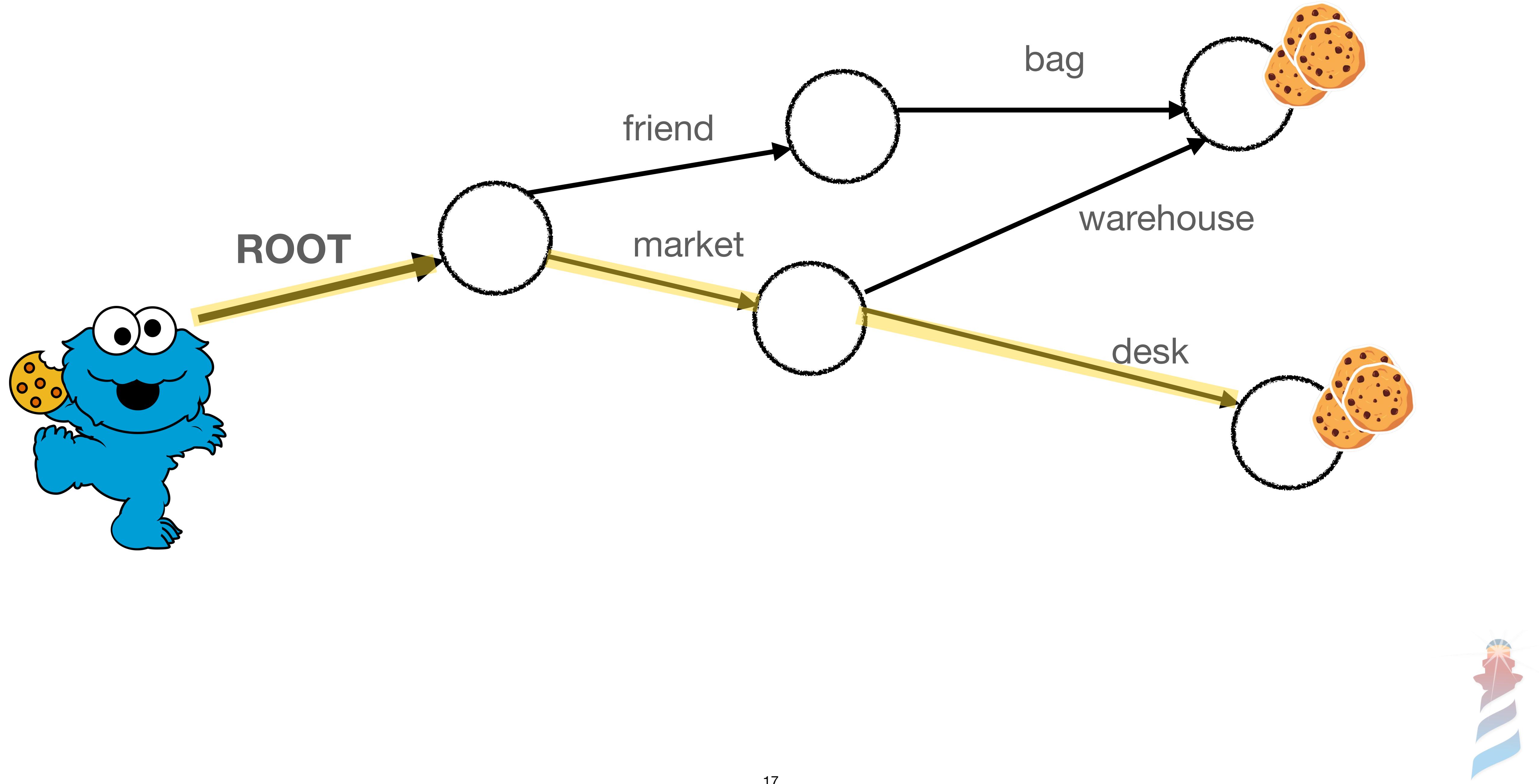


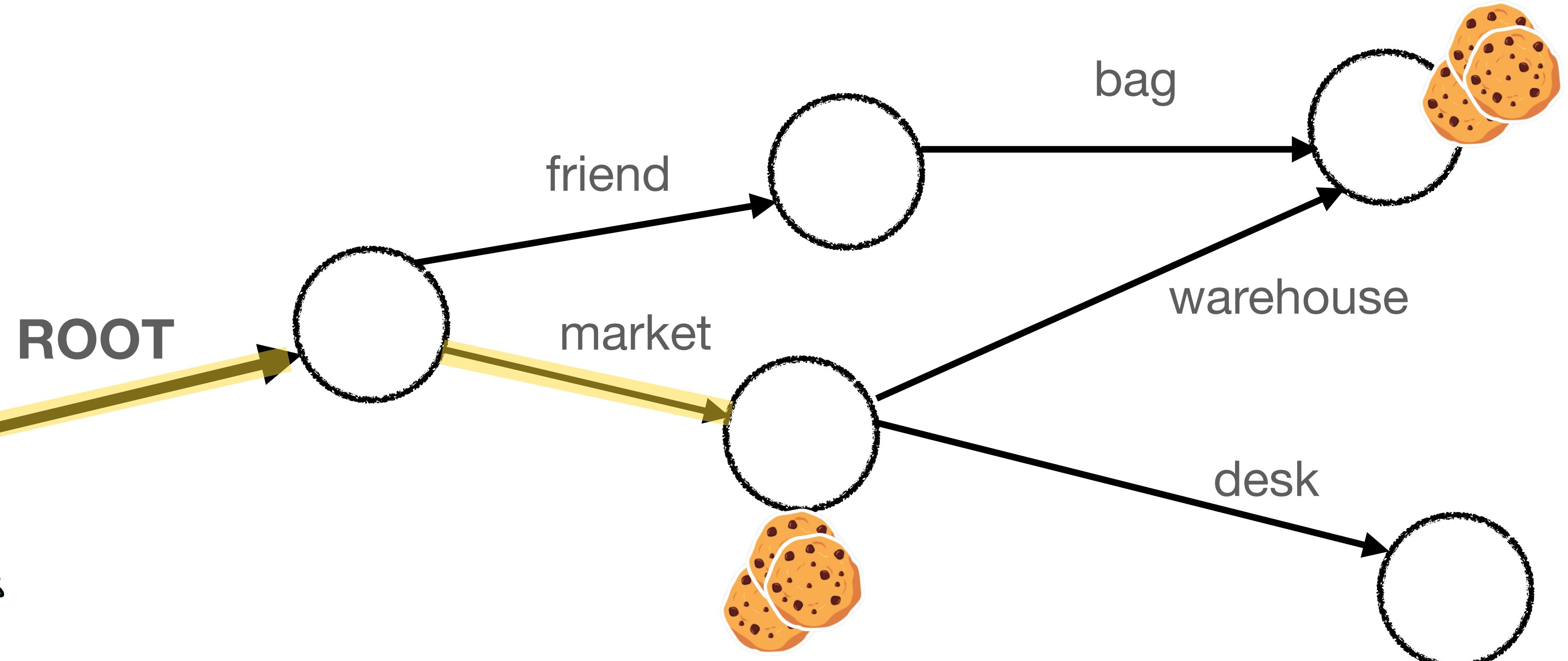


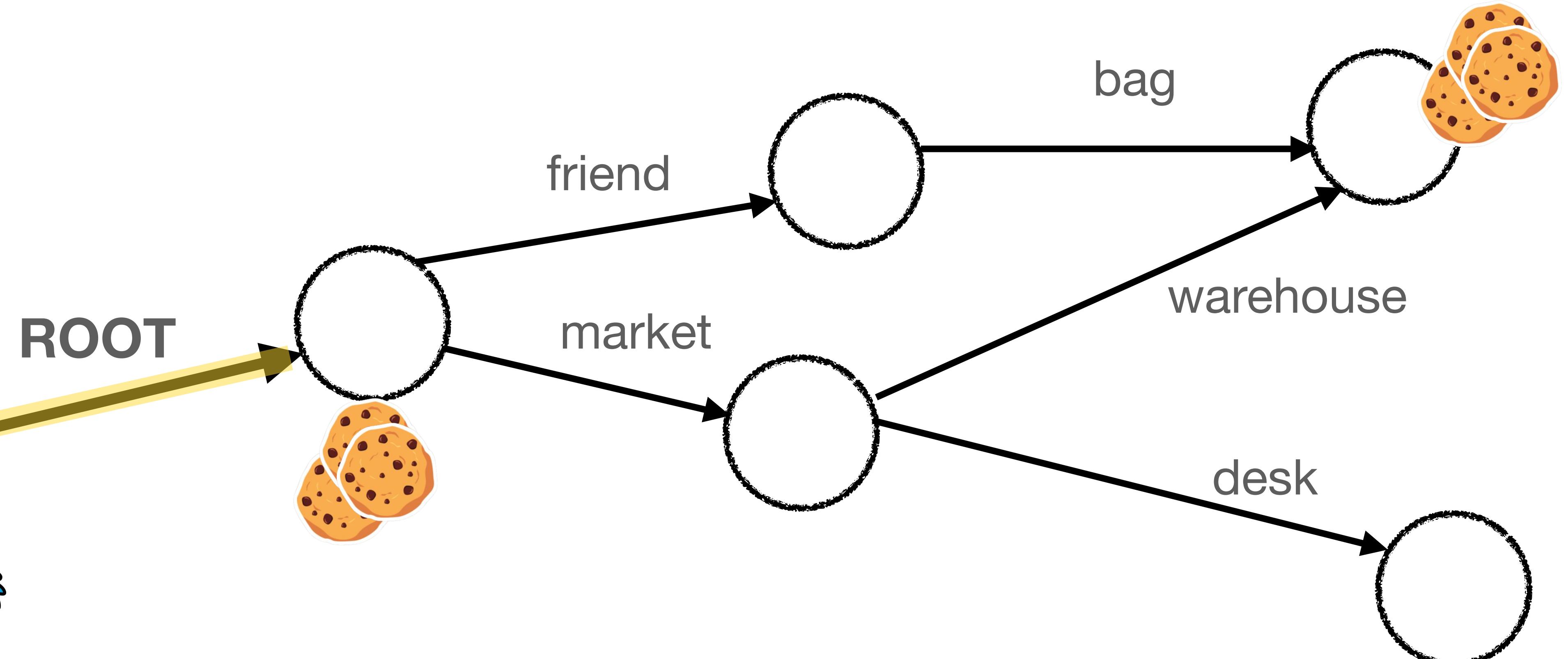


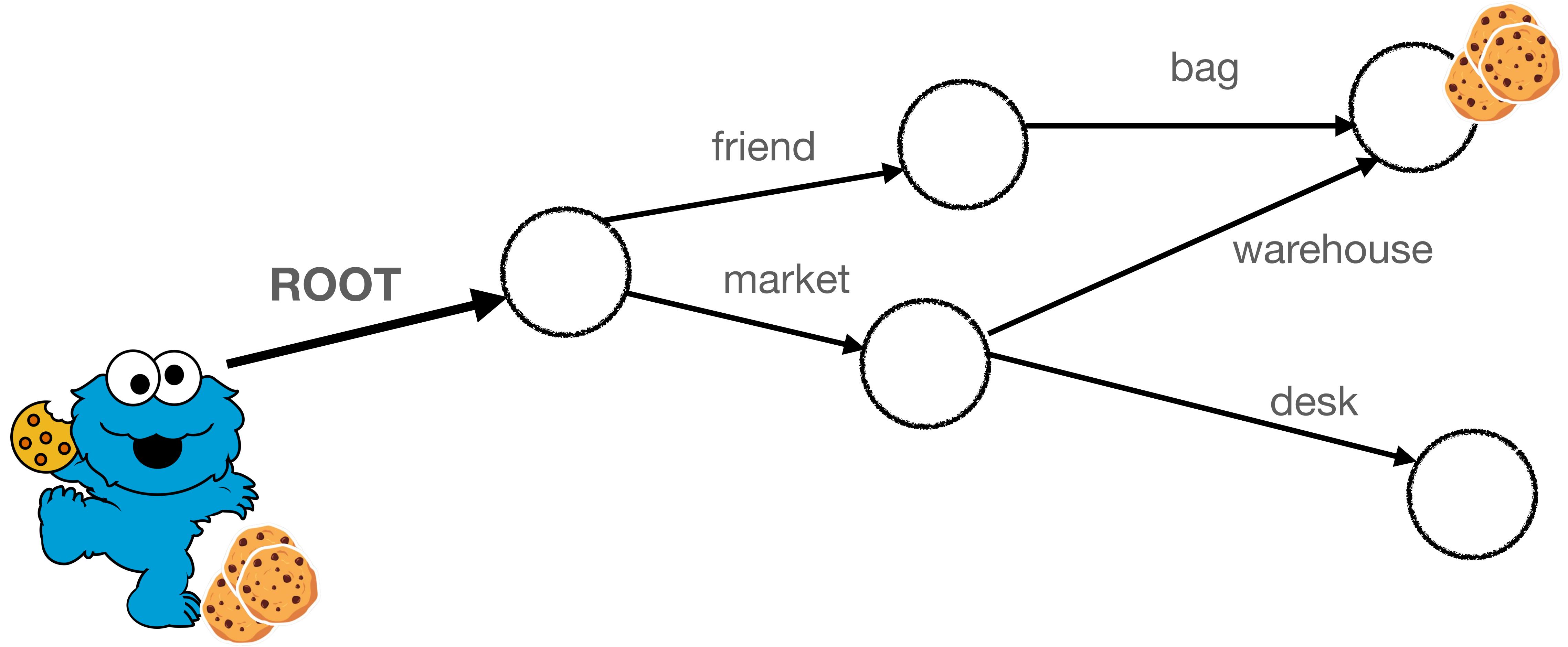


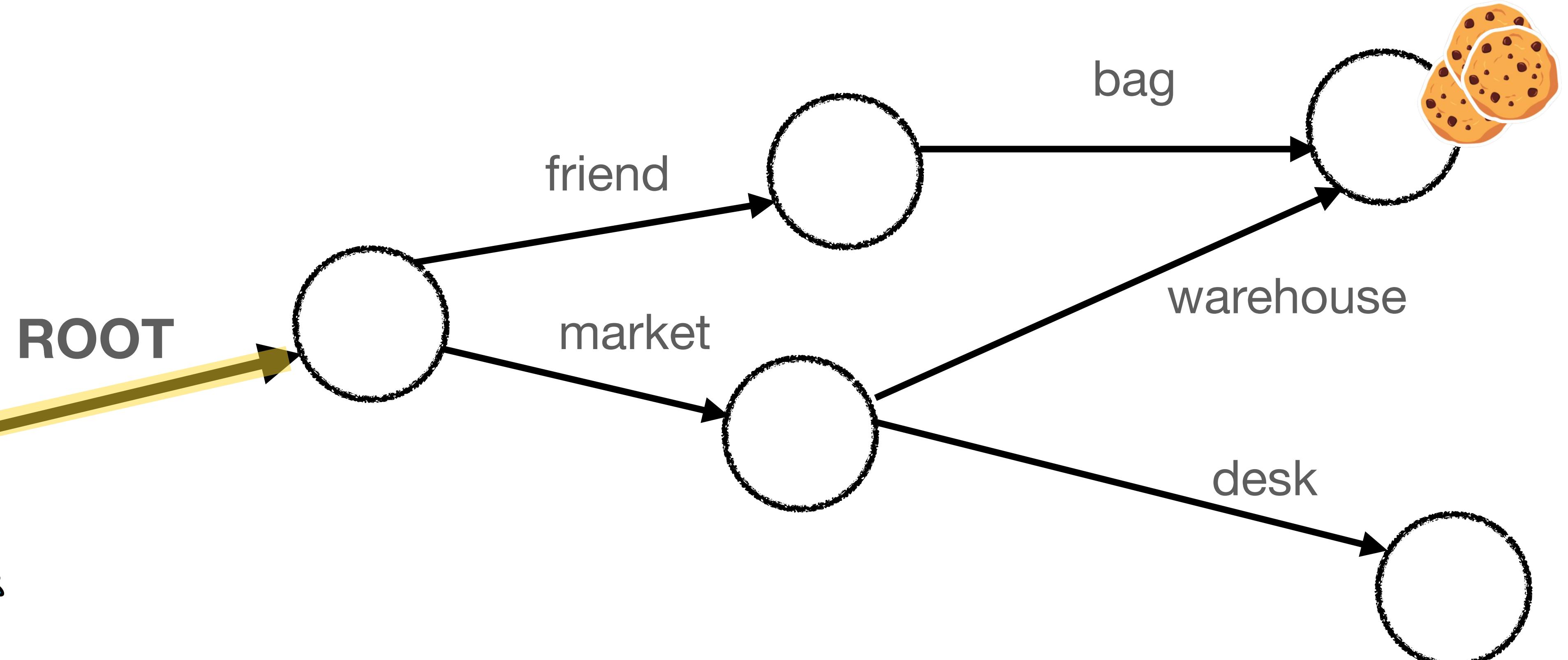


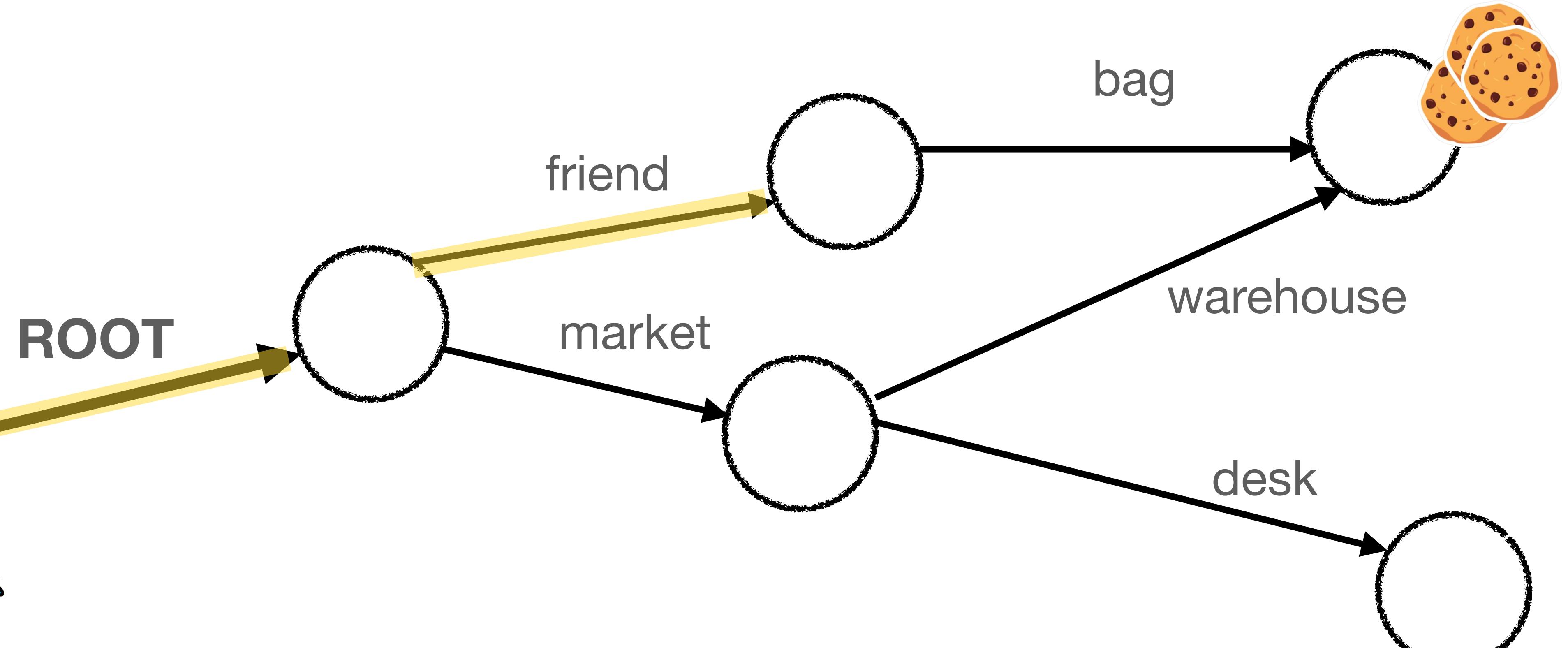


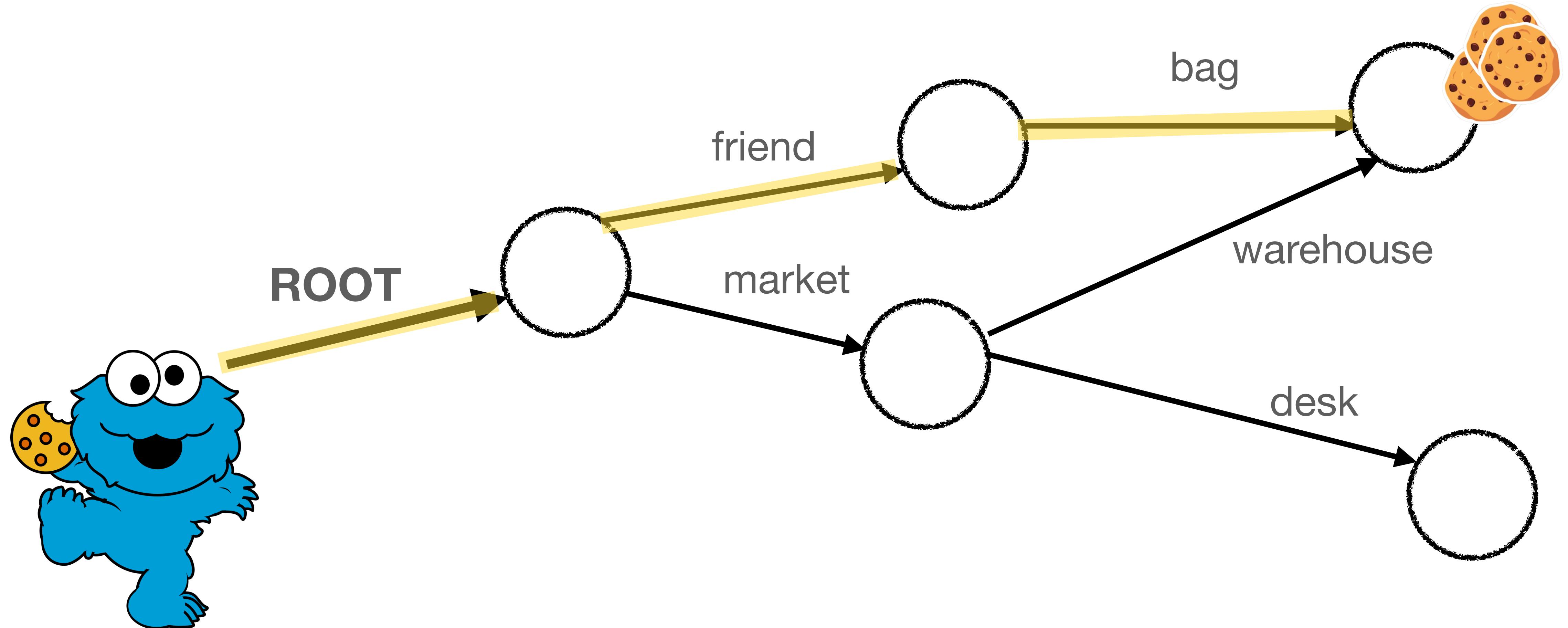














**ROOT**

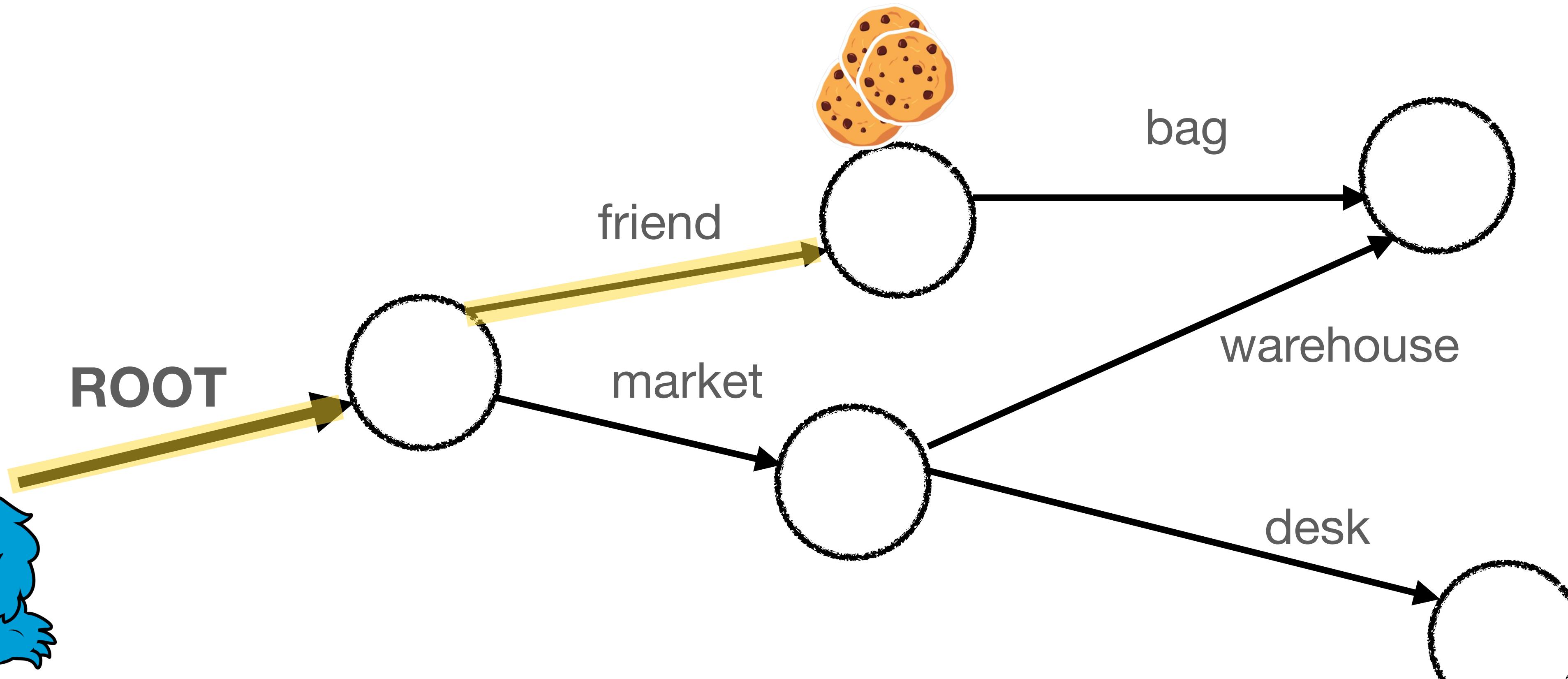
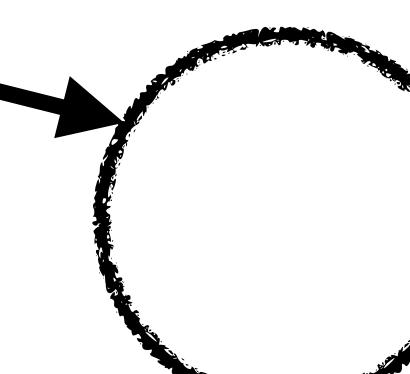
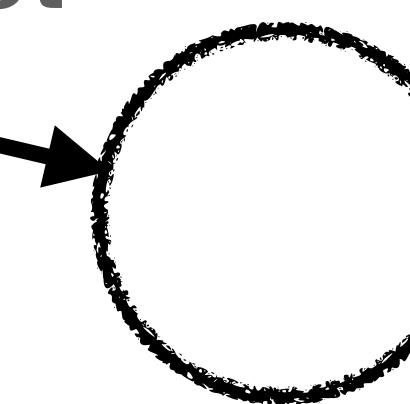
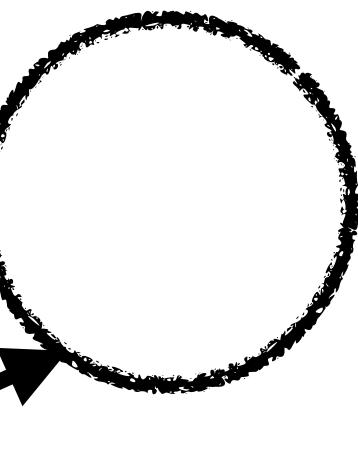
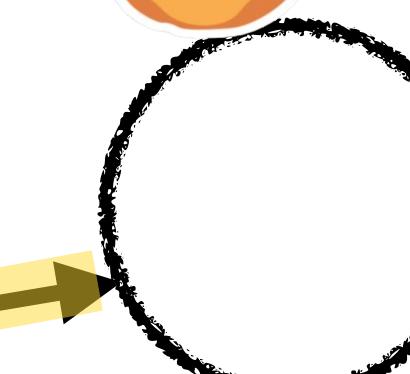
friend

market

bag

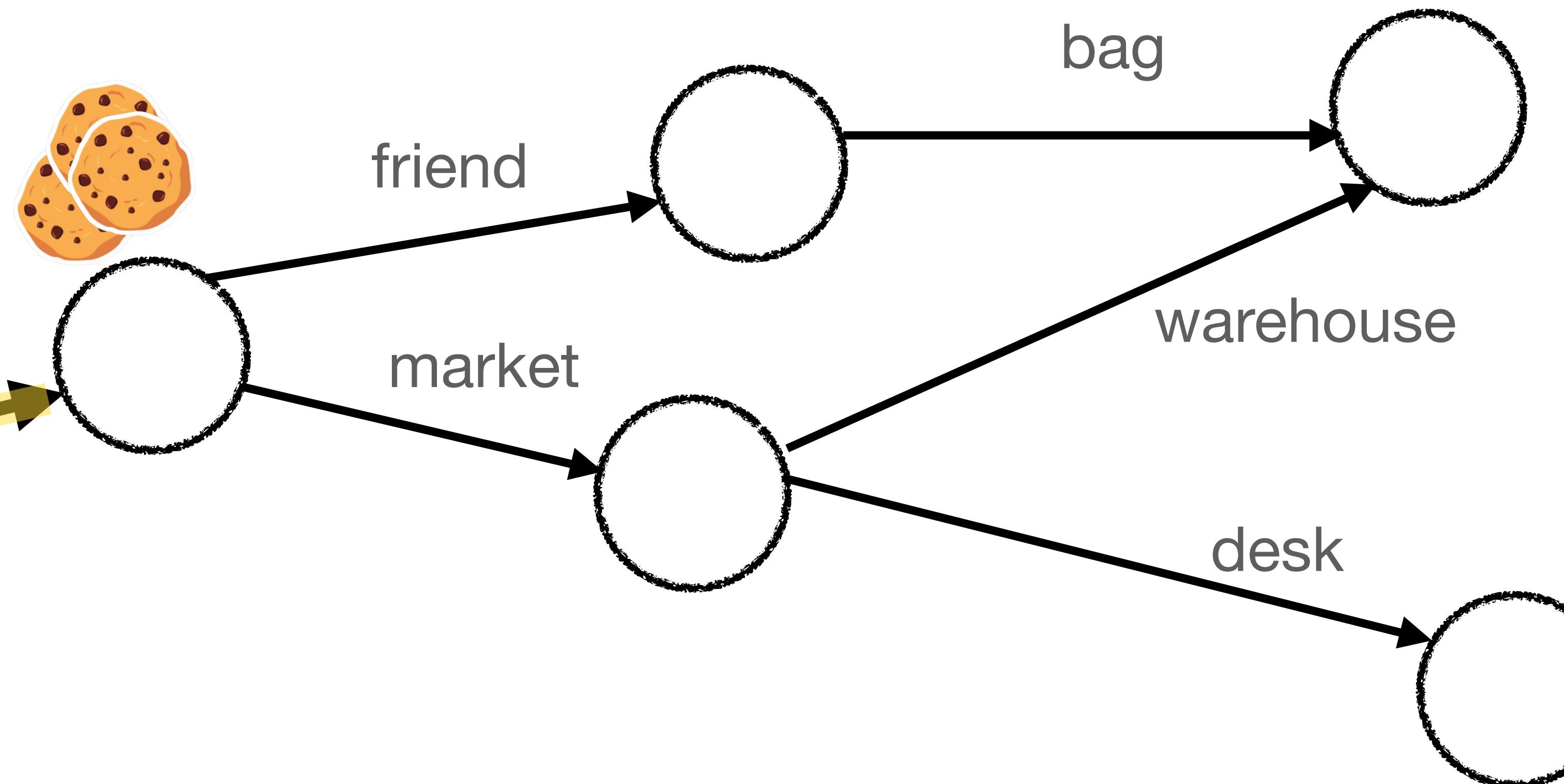
warehouse

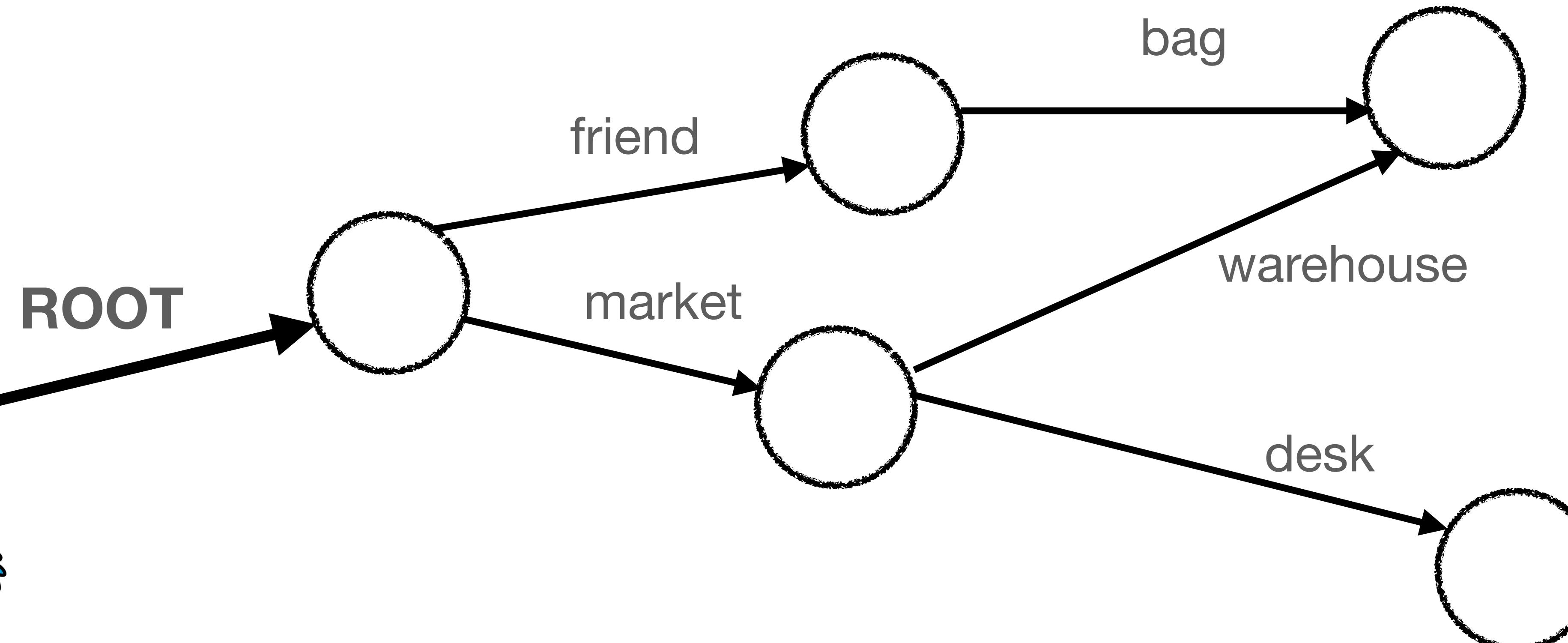
desk

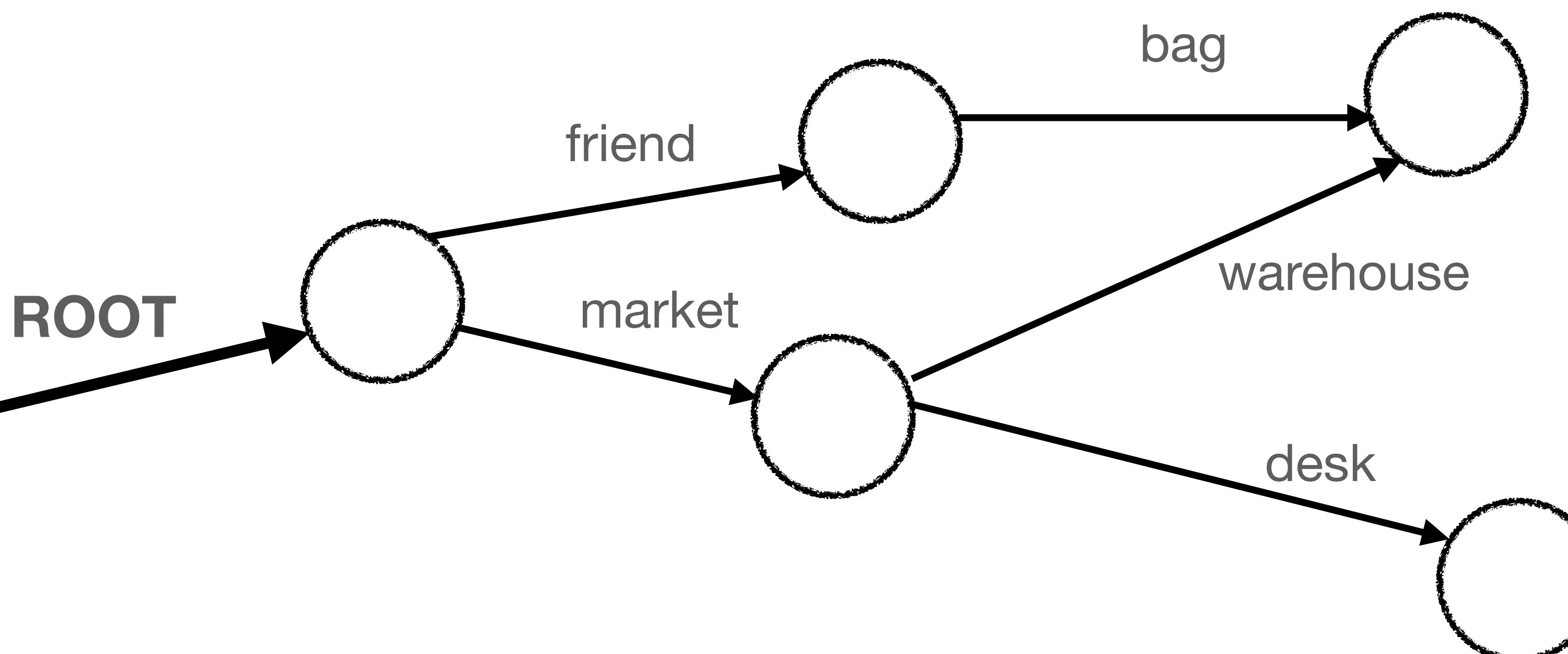




**ROOT**

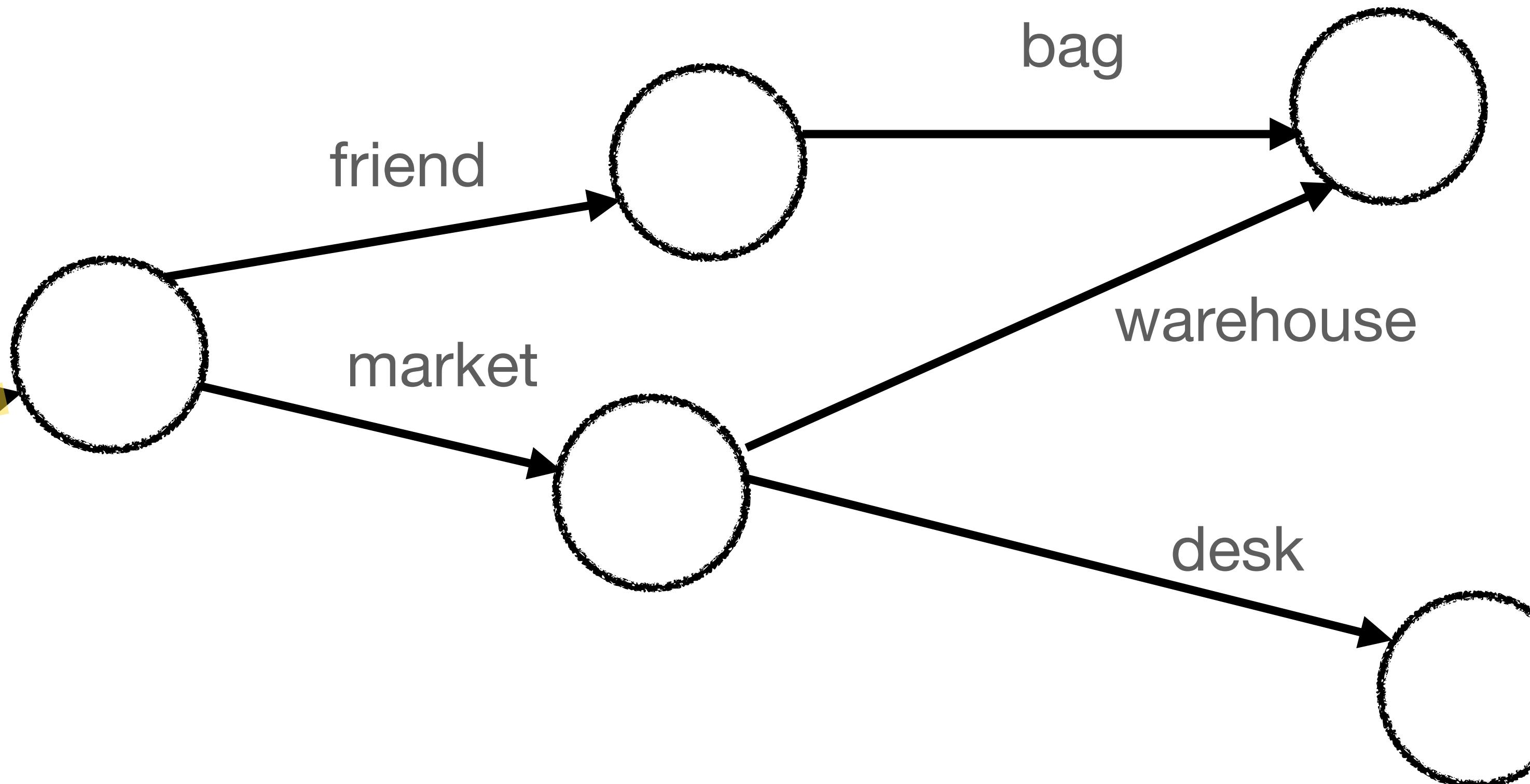






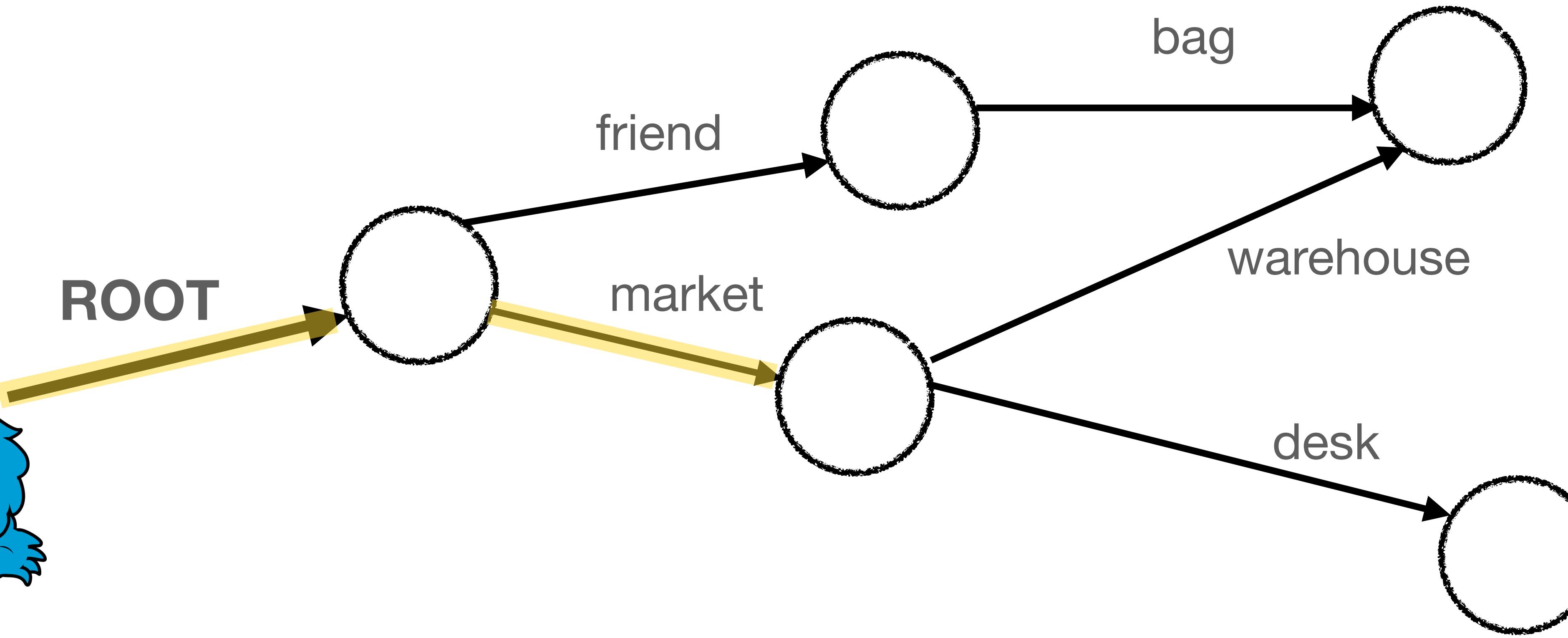


**ROOT**



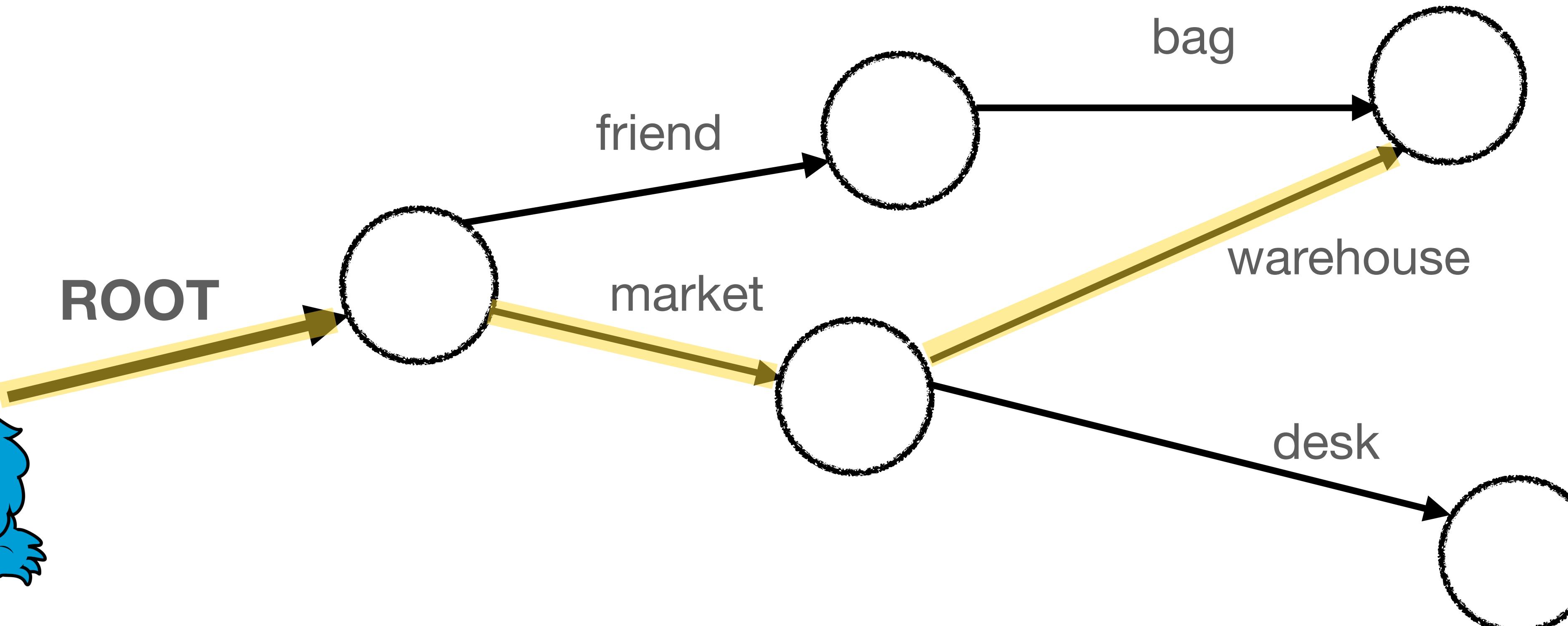


**ROOT**



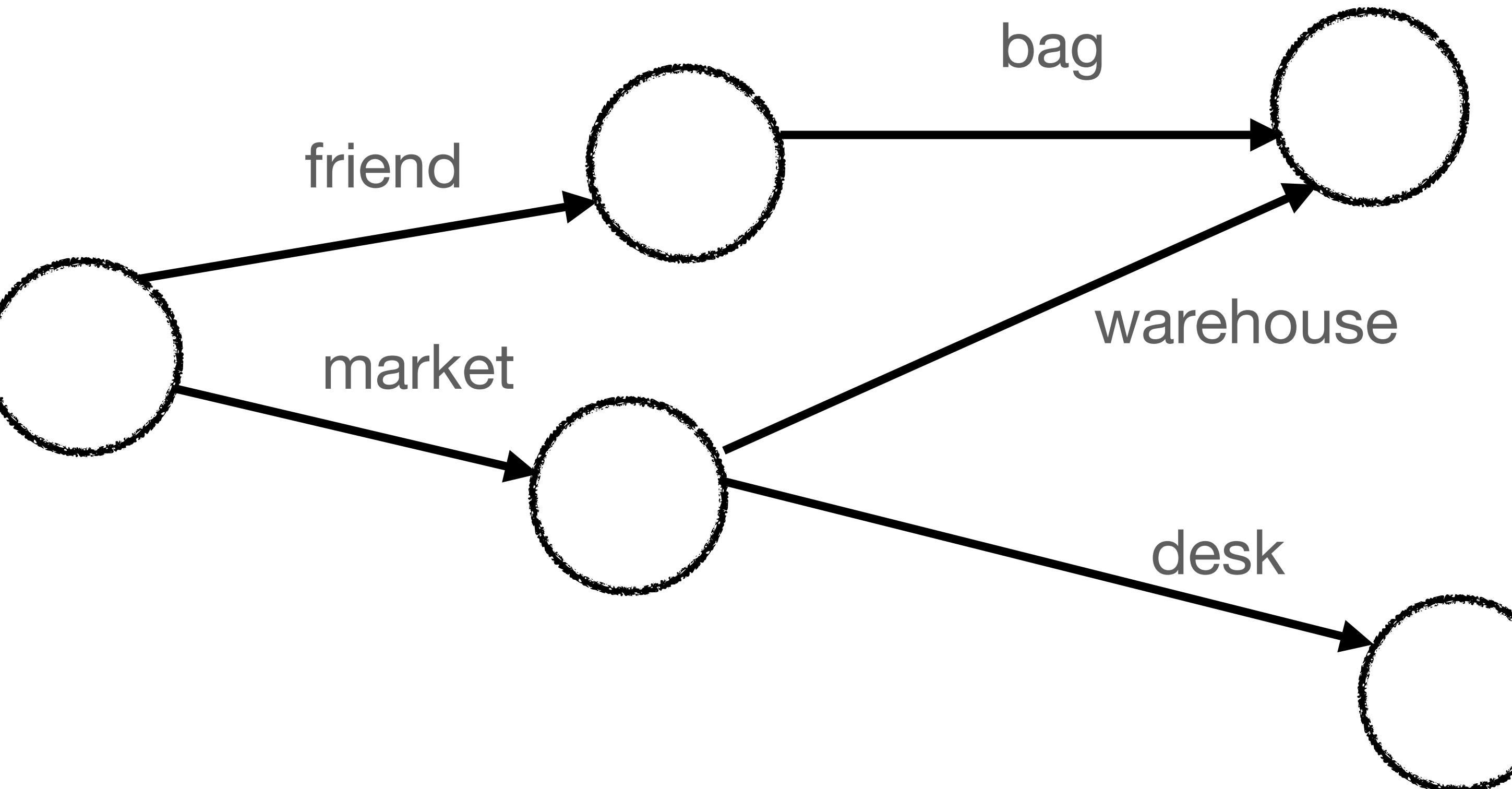


**ROOT**



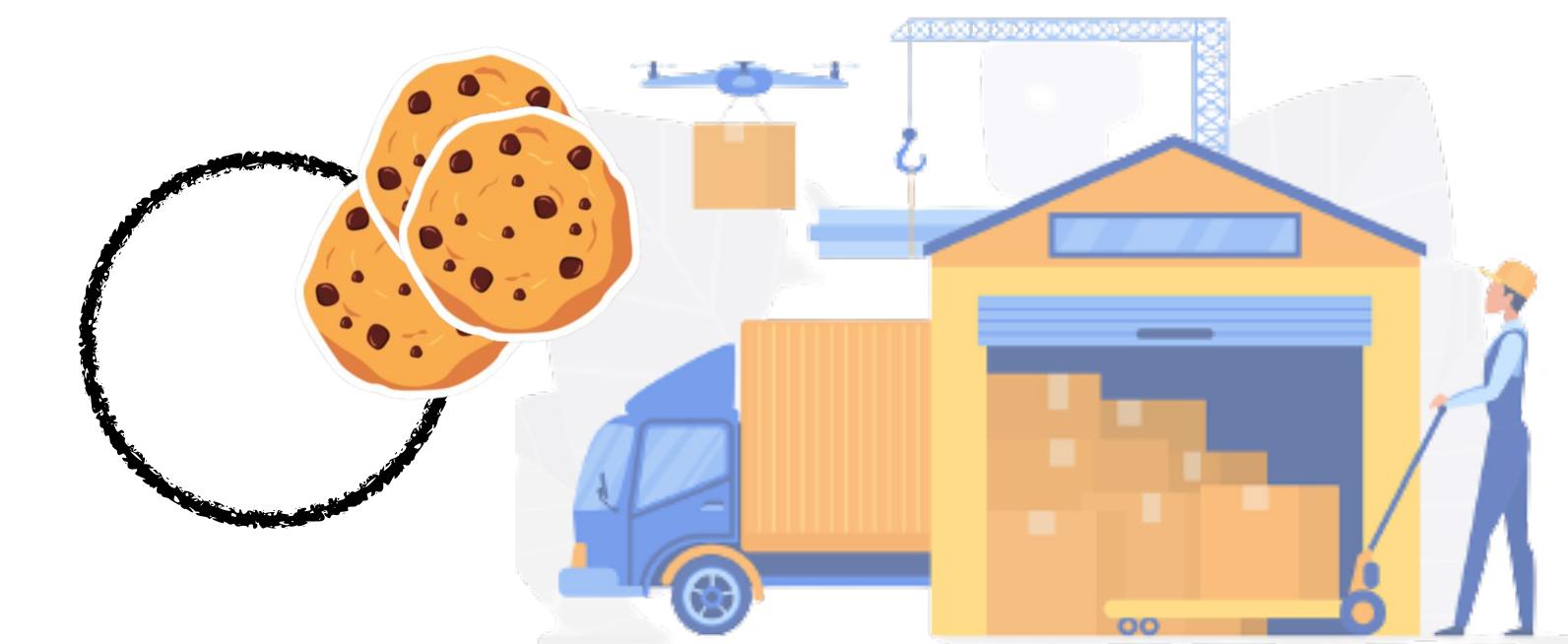
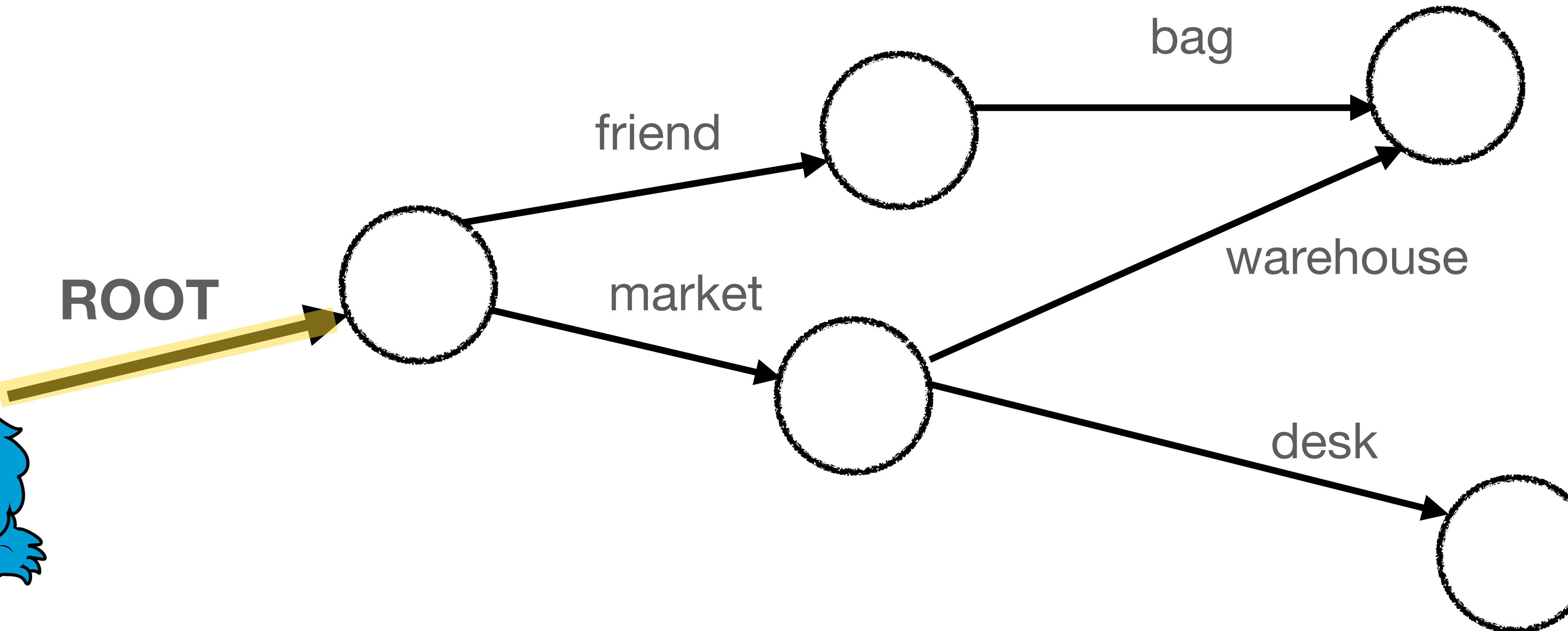


**ROOT**



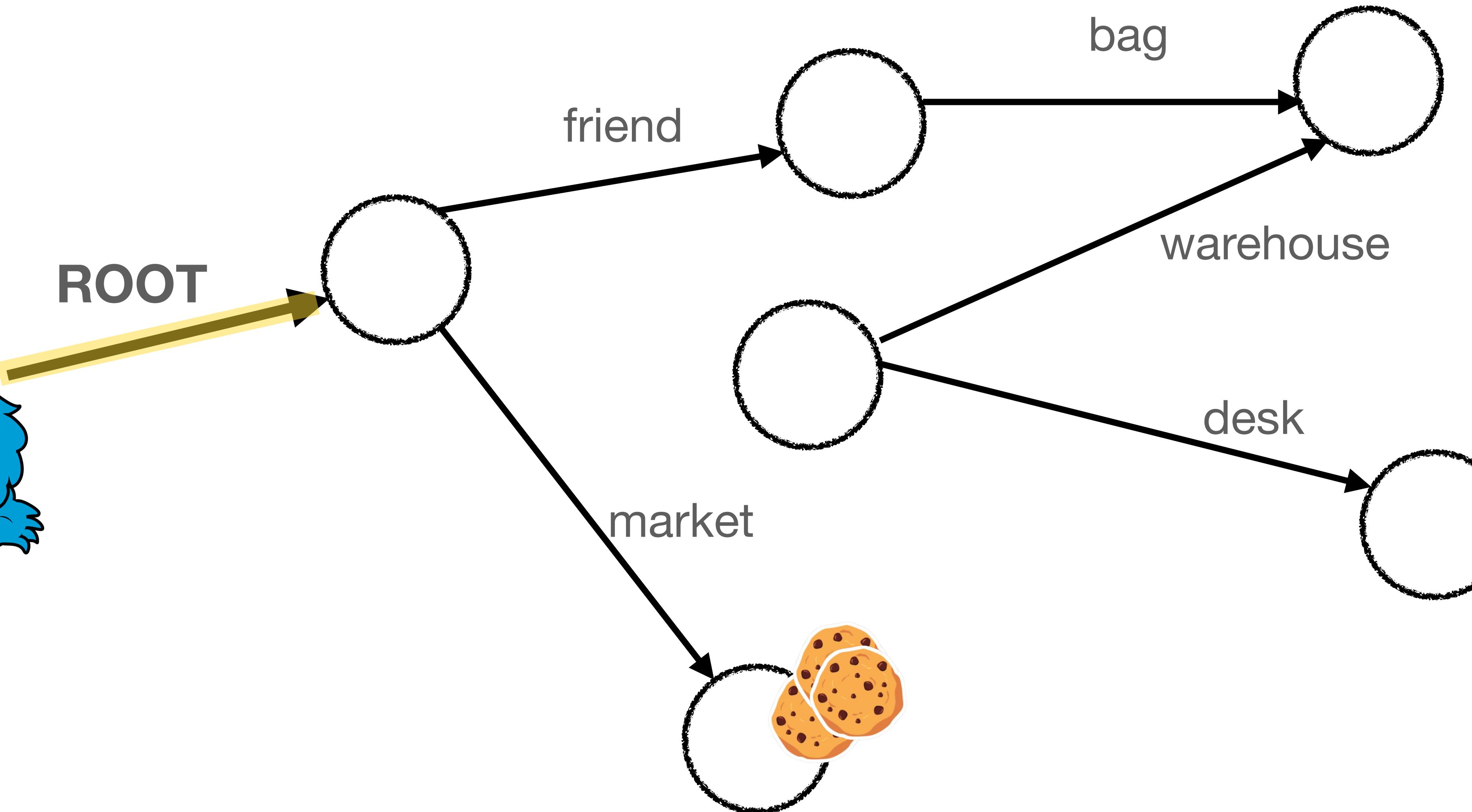


**ROOT**



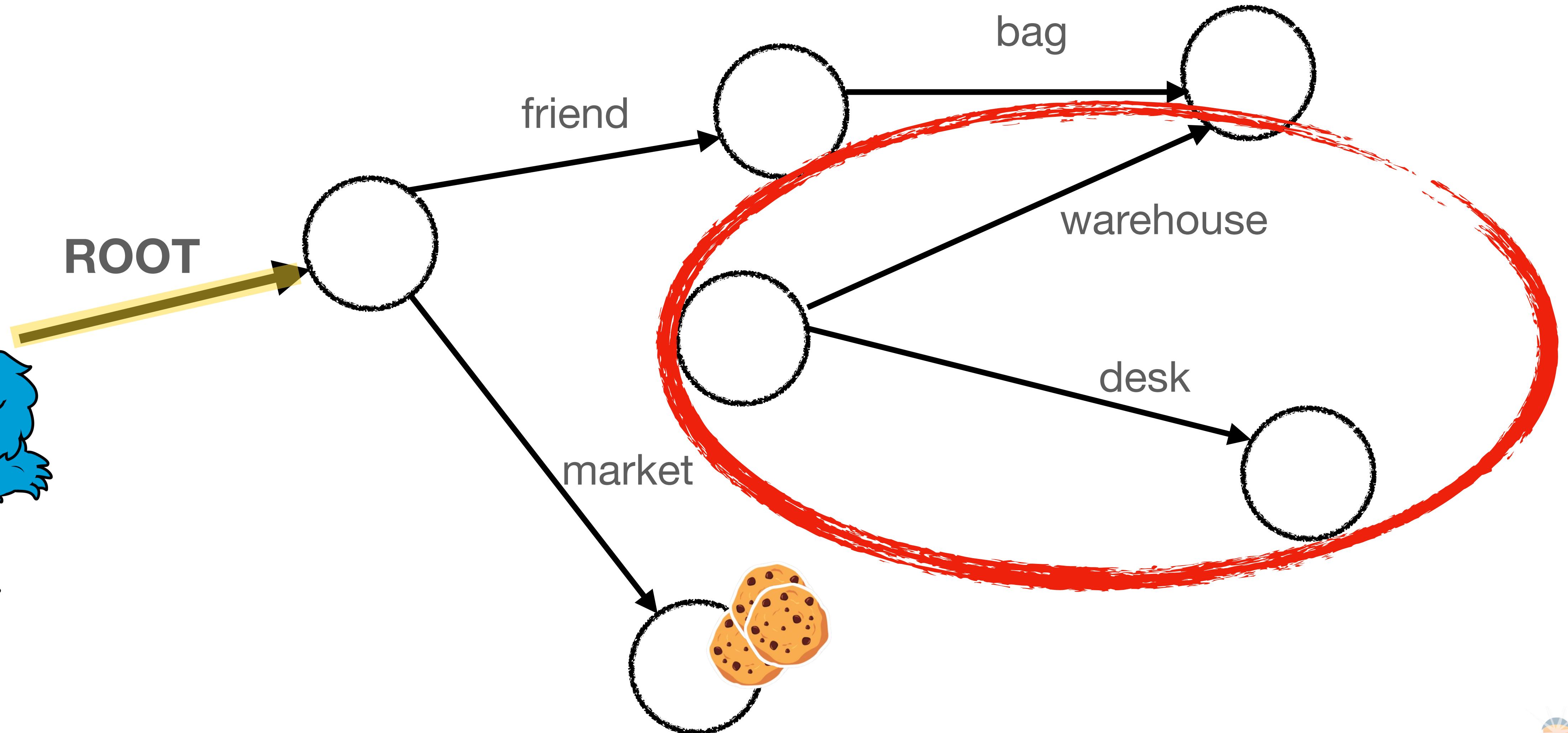


**ROOT**



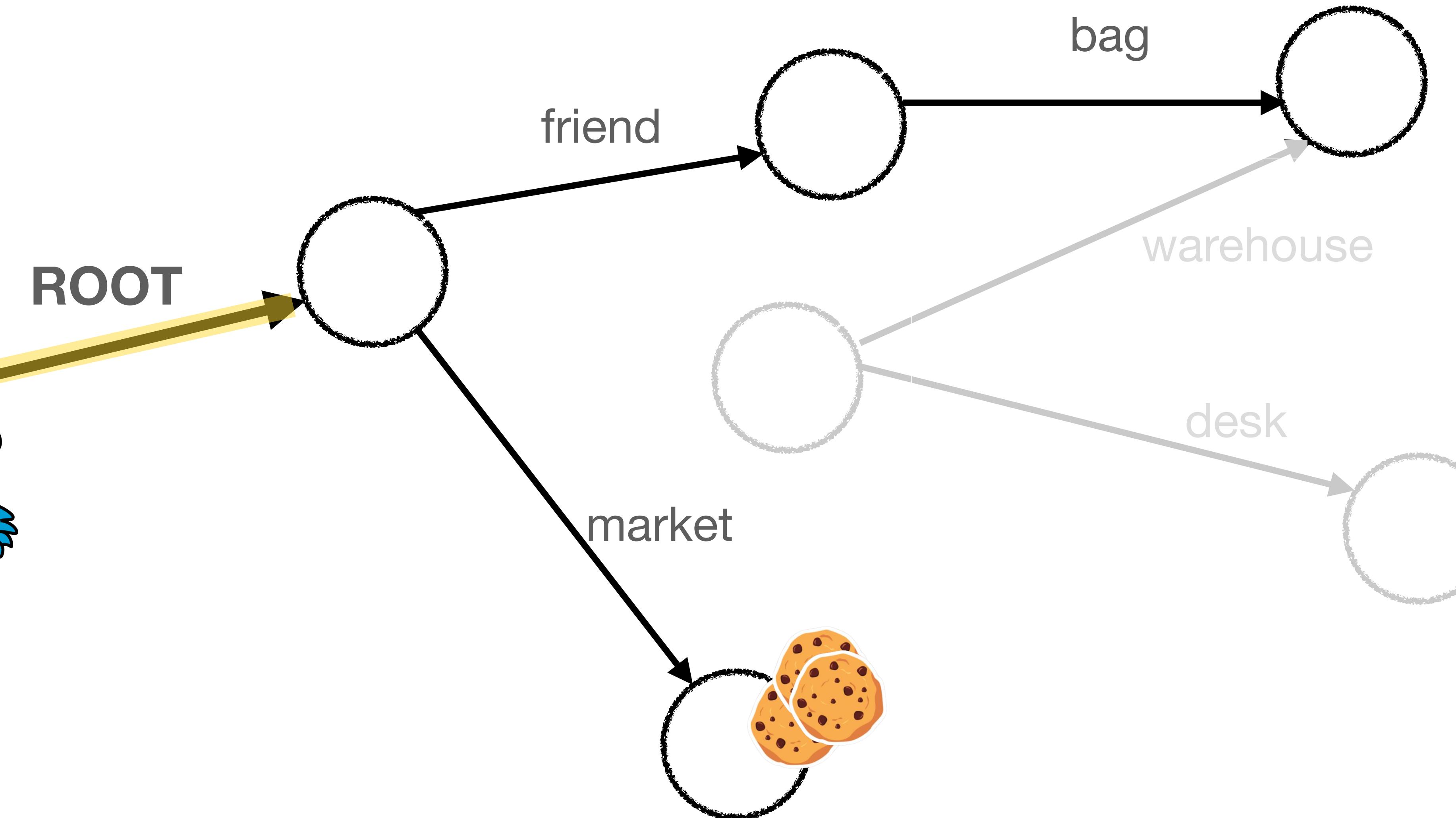


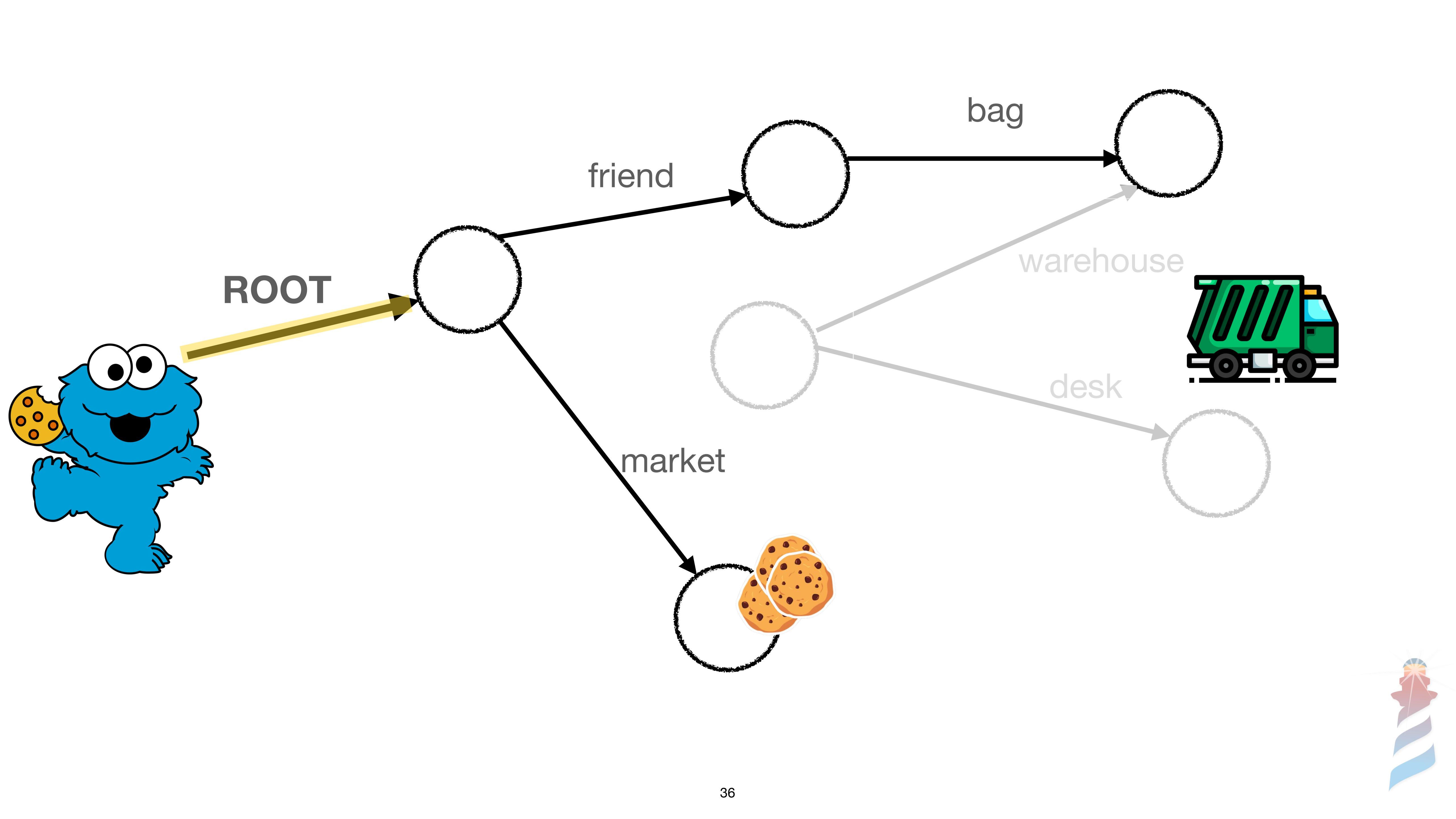
**ROOT**

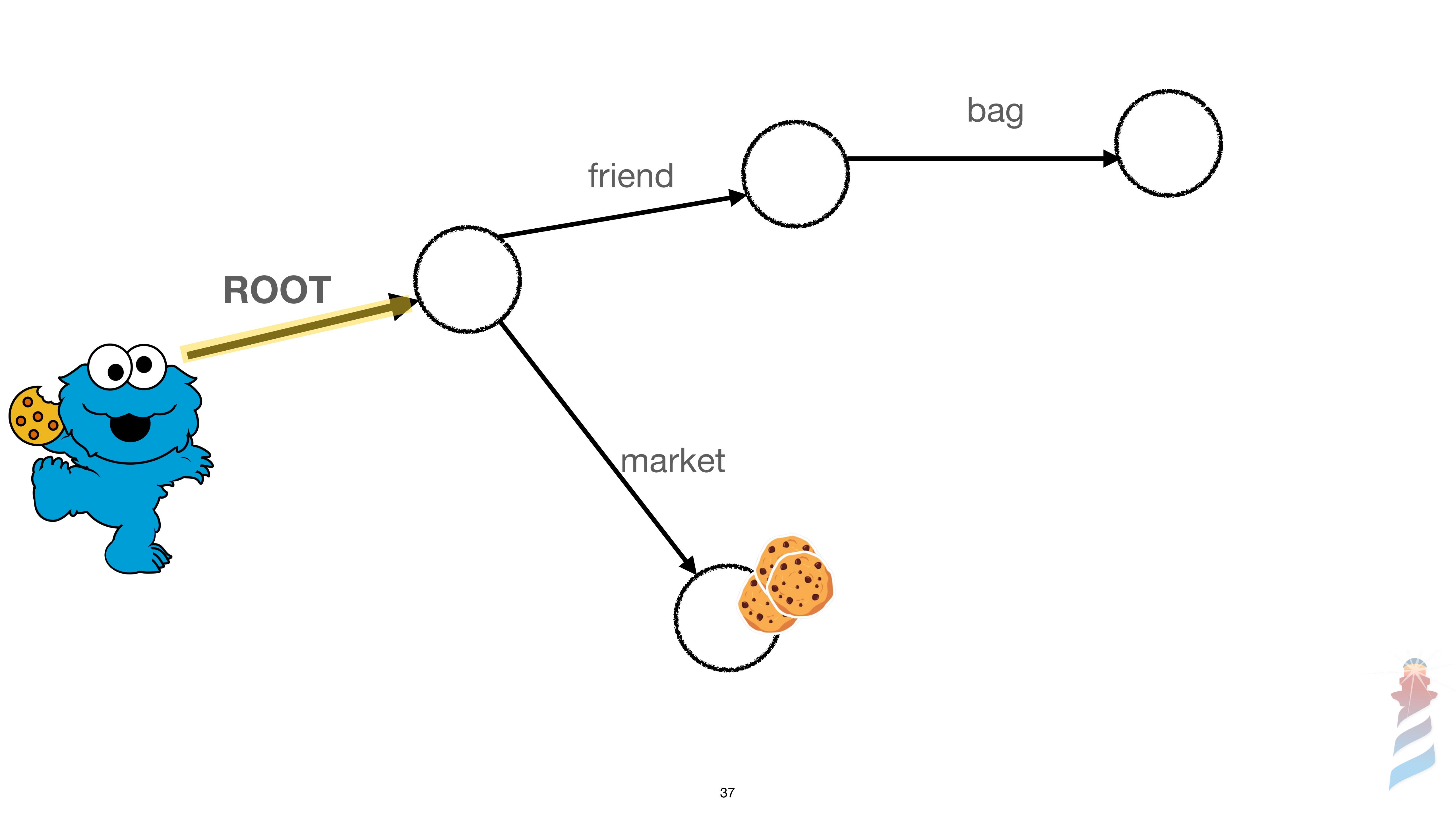


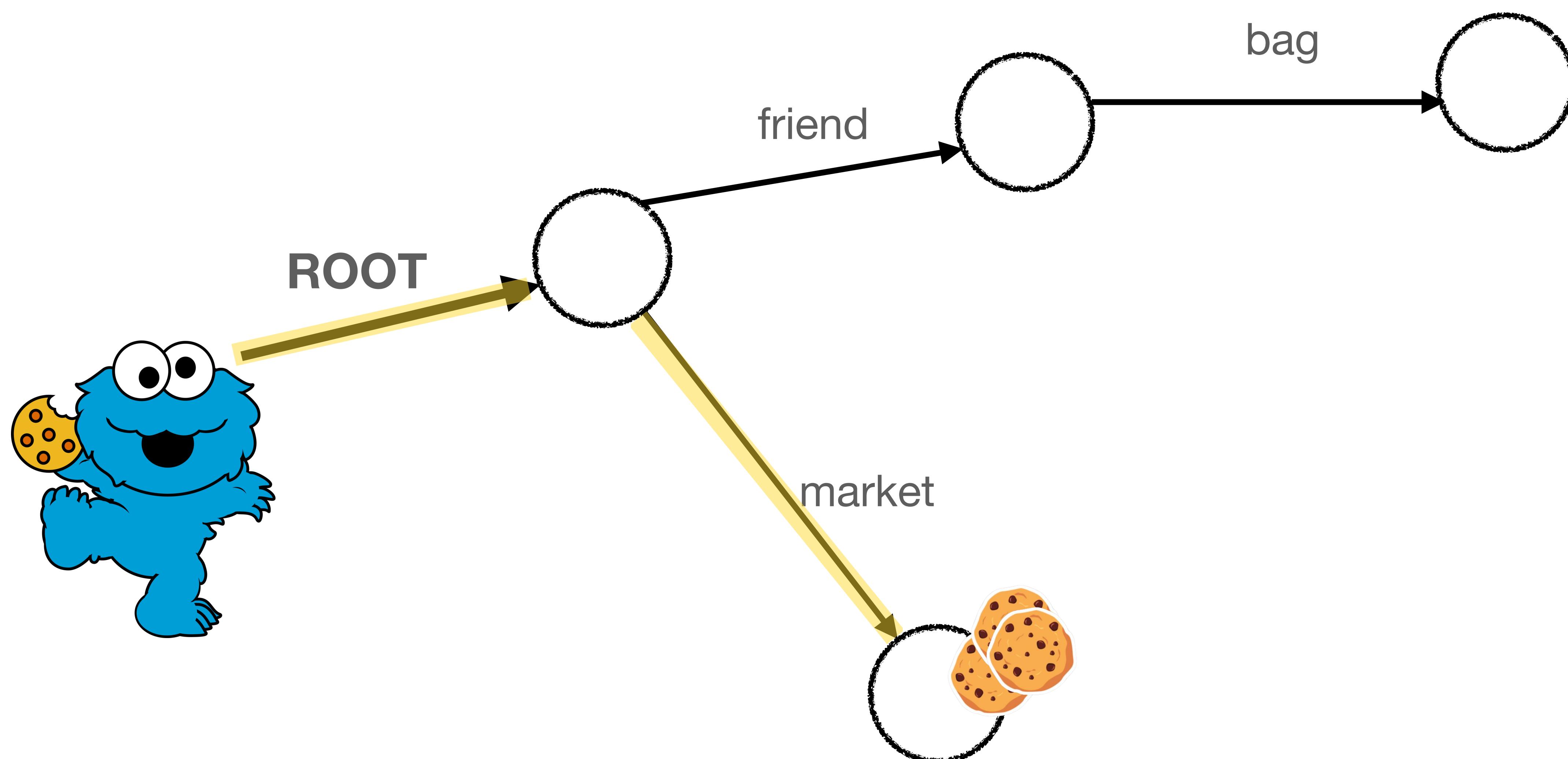


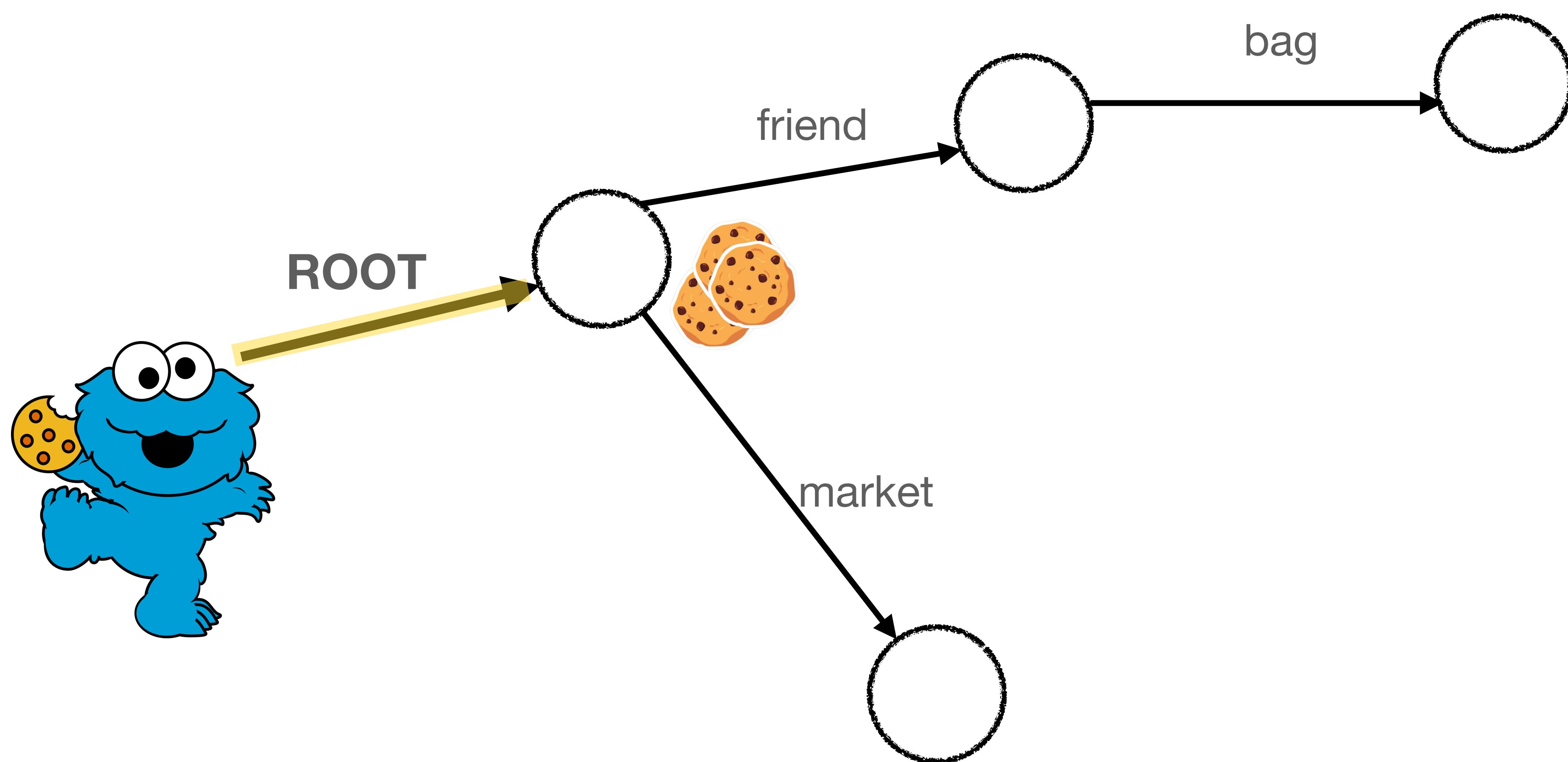
**ROOT**

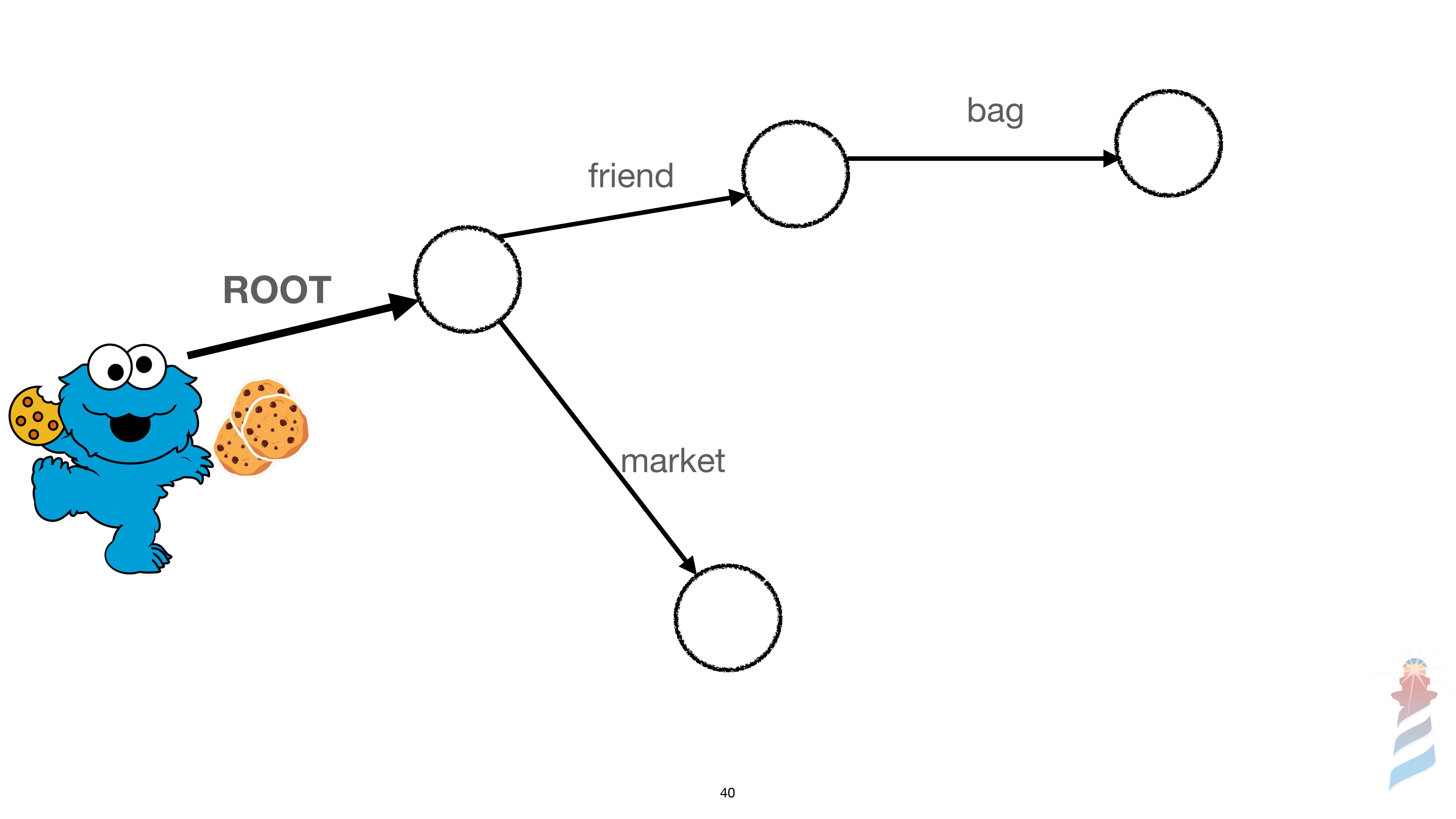












# Time to code



# Pharo

<https://pharo.org/download>

