OpenLambda Architecture
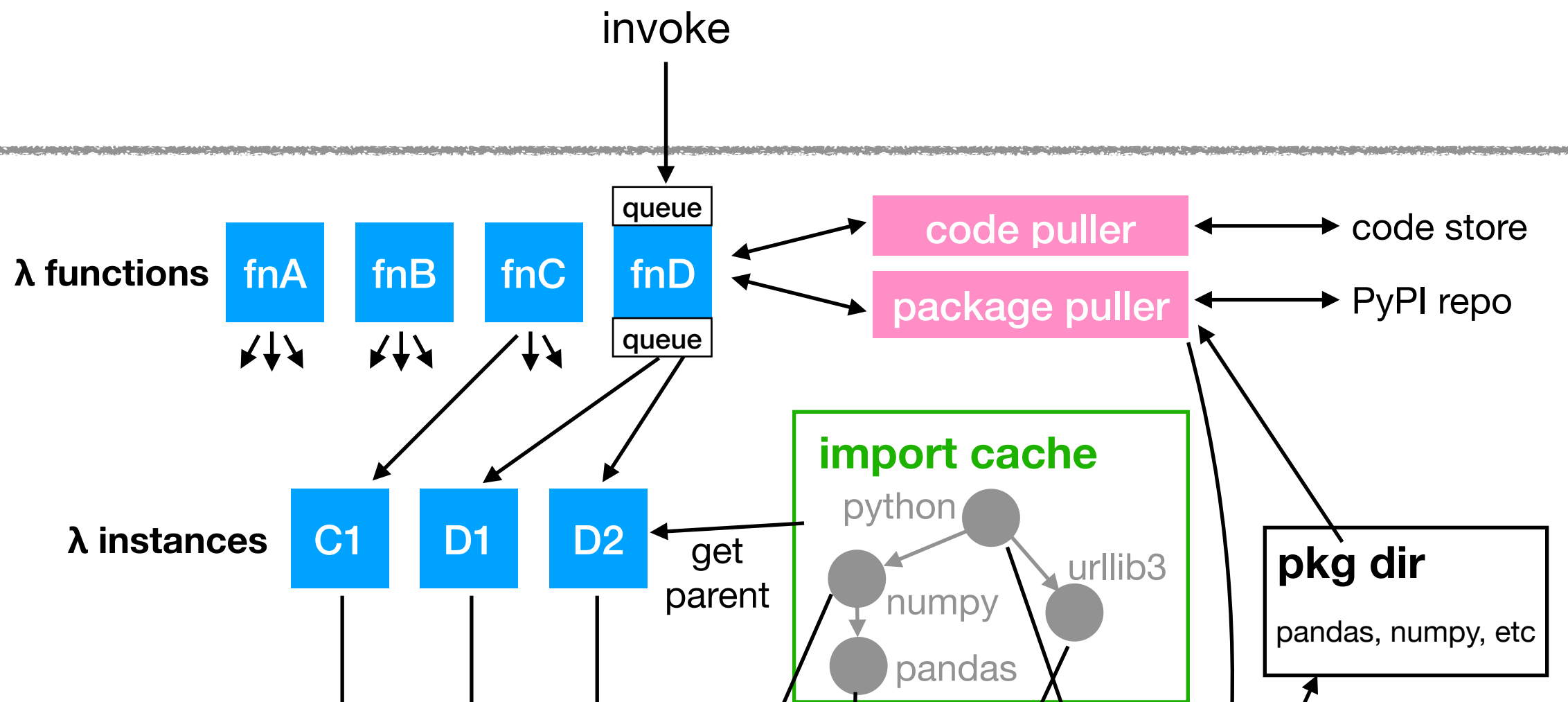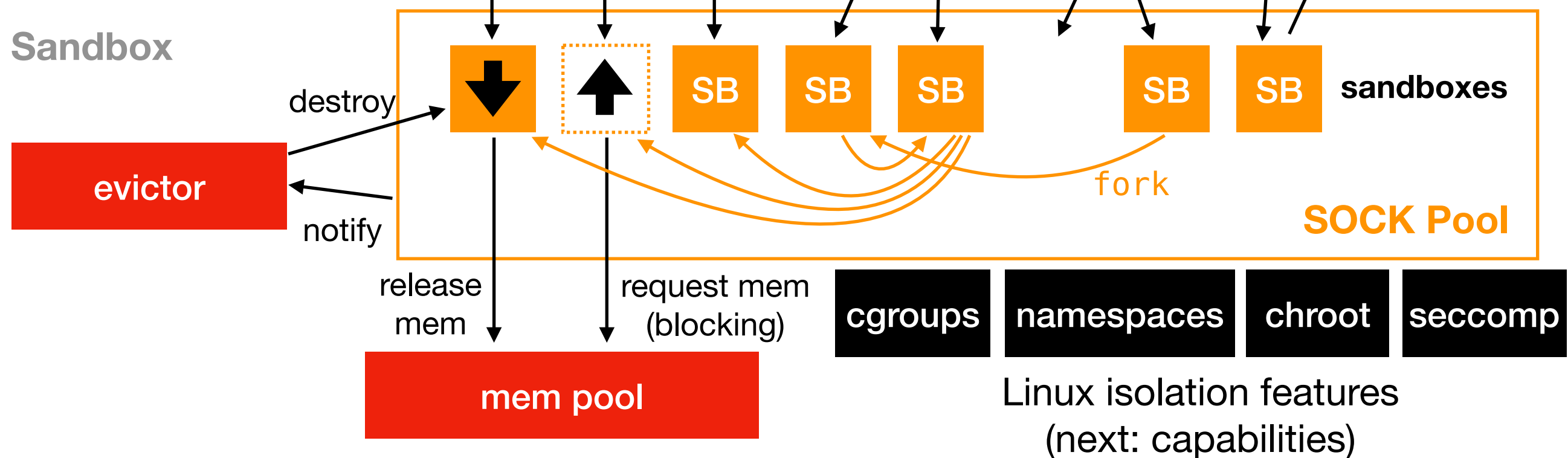
Tyler Caraza-Harter

Event

invoke

Lambda

λ functions   fnA   fnB   fnC   fnD   queue   code puller   code store

queue   package puller   PyPI repo

λ instances   C1   D1   D2   get parent

import cache
python
numpy   urllib3
pandas

pkg dir
pandas, numpy, etc

Sandbox

destroy   SB   SB   SB   SB   SB   sandboxes

fork

SOCK Pool

OpenLambda uses a custom container implementation (SOCK) instead of Docker optimized for Lambdas, but which could be used for other systems.

mem
king)

mem pool

cgroups   namespaces   chroot   seccomp

Linux isolation features
(next: capabilities)

Event

invoke

Lambda

λ functions

fnA fnB fnC fnD queue

code puller ←→ code store

package puller ←→ PyPI repo

queue

import cache

python

numpy

urllib3

pandas

λ instances

C1 D1 D2 get parent

pkg dir

pandas, numpy, etc

Sandbox

destroy

SB SB SB SB SB sandboxes

fork

SOCK Pool
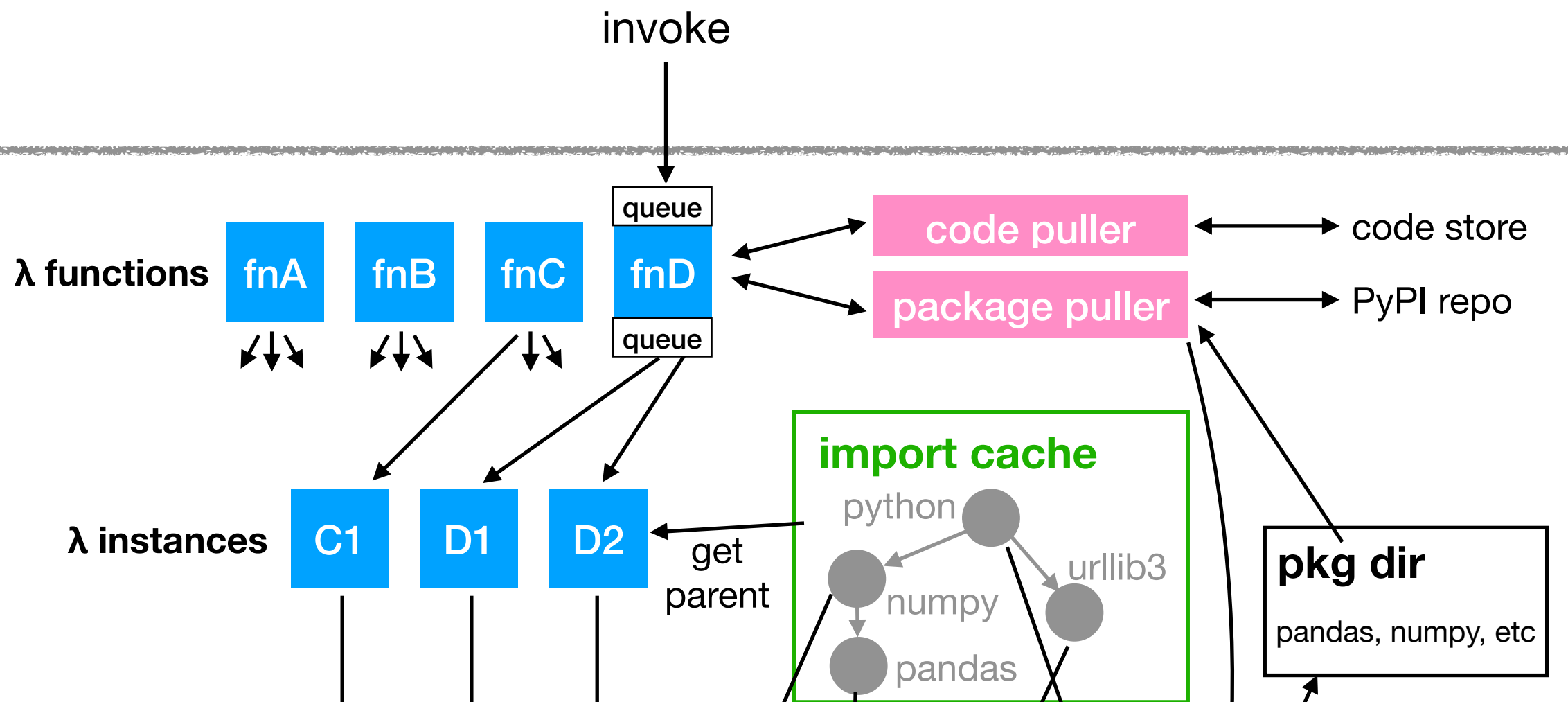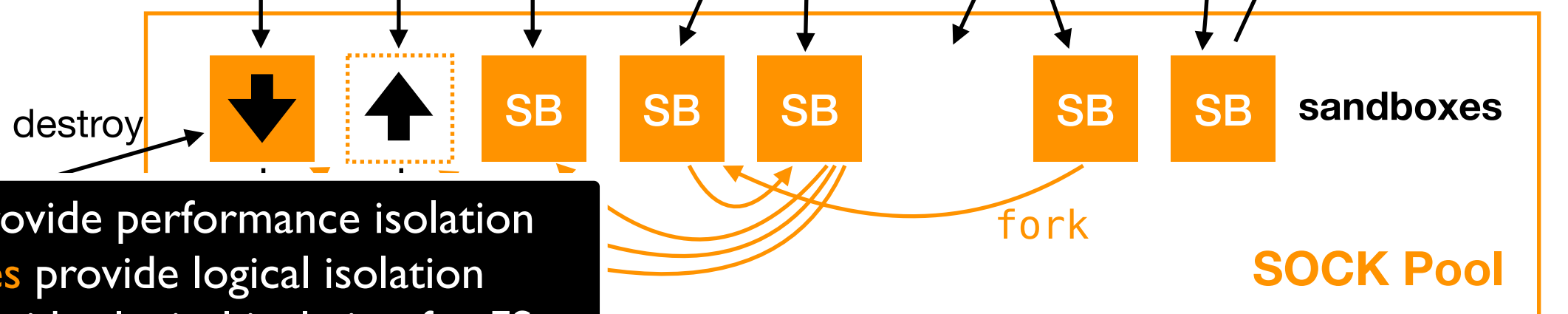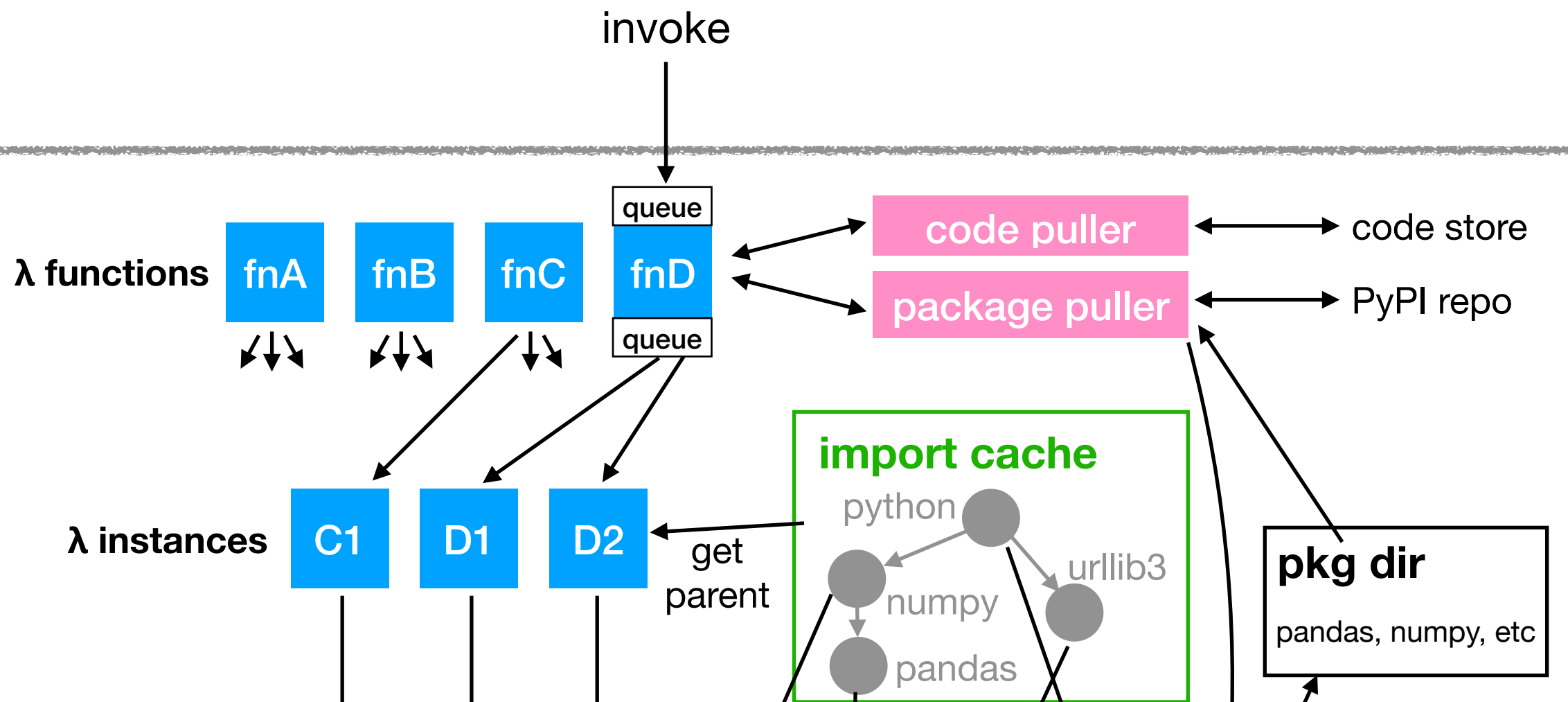
- cgroups provide performance isolation
- namespaces provide logical isolation
- chroot provides logical isolation for FS
- seccomp blocks "dangerous" syscalls

mem pool

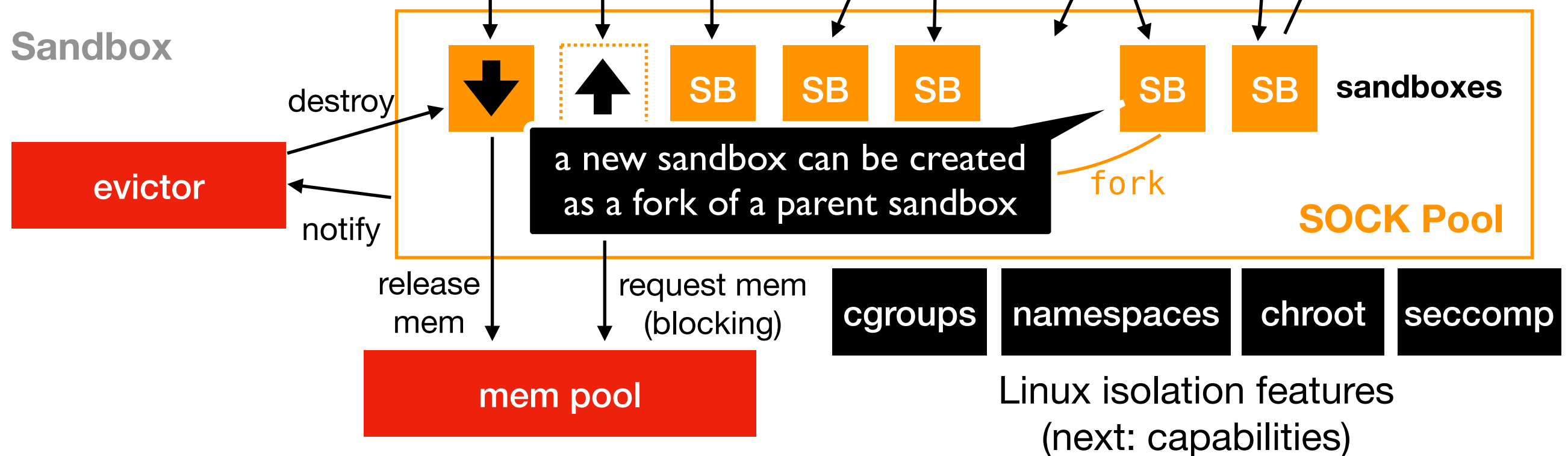cgroups namespaces chroot seccomp

Linux isolation features
(next: capabilities)

Event

invoke

Lambda

λ functions

fnA fnB fnC queue fnD queue

code puller ⟷ code store

package puller ⟷ PyPI repo

λ instances

C1 D1 D2

get parent

import cache

python

numpy

pandas

urllib3

pkg dir

pandas, numpy, etc

Sandbox

destroy

SB SB SB SB SB **sandboxes**

evictor

notify

fork

a new sandbox can be created as a fork of a parent sandbox

SOCK Pool

release mem

request mem (blocking)

mem pool

cgroups namespaces chroot seccomp

Linux isolation features (next: capabilities)

```
type SandboxPool interface {
  Create(parent Sandbox, isLeaf bool, codeDir, ...) (Sandbox, error)
...
}
```
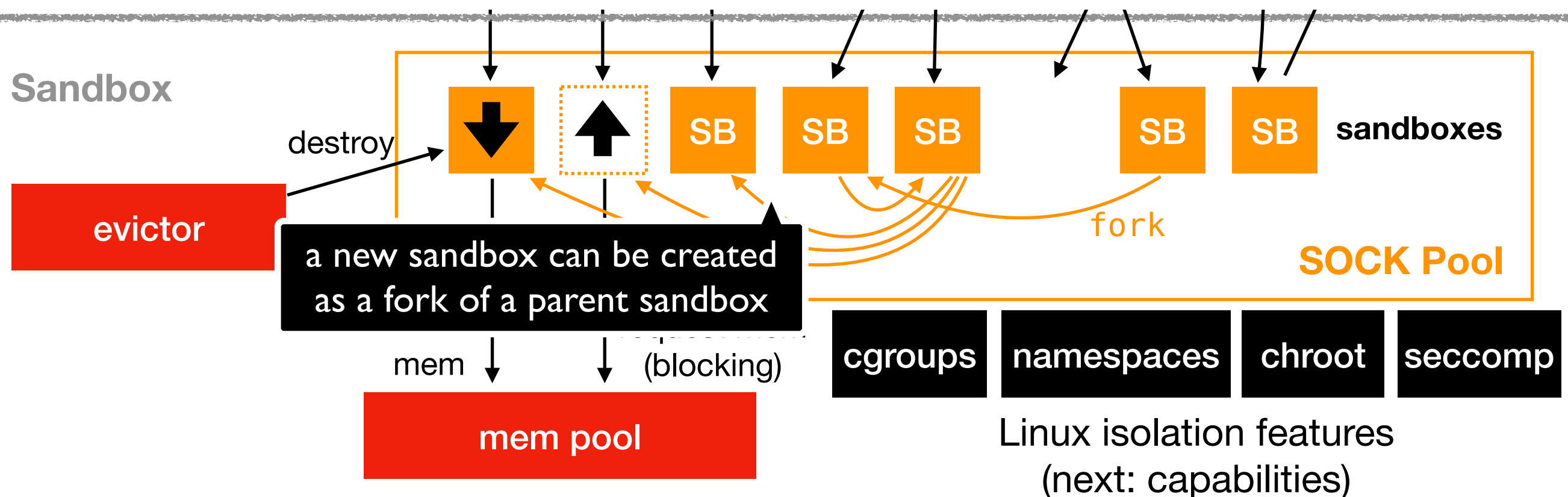
where to fork from
(may be null)

is this a handler (true)
or a Zygote (false)

path to Lambda code

Sandbox API

Sandbox

destroy

SB   **SB**   **SB**   **SB**   **SB**   **sandboxes**

evictor

a new sandbox can be created
as a fork of a parent sandbox

fork

SOCK Pool

mem   (blocking)

mem pool

cgroups   namespaces   chroot   seccomp

Linux isolation features
(next: capabilities)

```go
type SandboxPool interface {
  Create(parent Sandbox, isLeaf bool, codeDir, ...) (Sandbox, error)
  ...
}
```

where to fork from
(may be null)

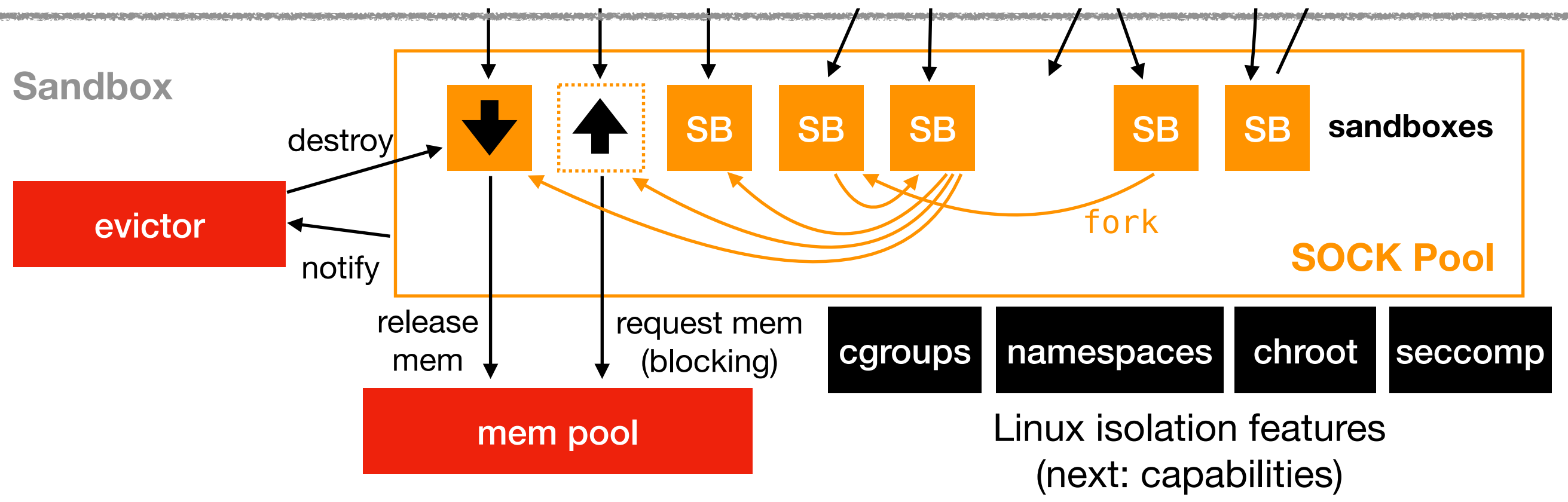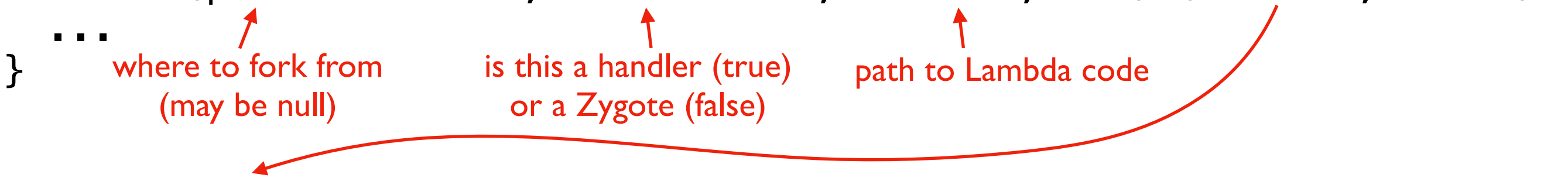is this a handler (true)
or a Zygote (false)

path to Lambda code

```go
type Sandbox interface {
  Client() (*http.Client)
  Pause() error
  Unpause() error
  ...
  Destroy(reason string)
  DestroyIfPaused(reason string)
}
```

```go
type SandboxPool interface {
  Create(parent Sandbox, isLeaf bool, codeDir, ...) (Sandbox, error)
  ...
}
```

where to fork from
(may be null)

is this a handler (true)
or a Zygote (false)

path to Lambda code

```go
type Sandbox interface {
  Client() (*http.Client)
  Pause() error
  Unpause() error
  ...
  Destroy(reason string)
  DestroyIfPaused(reason string)
}
```
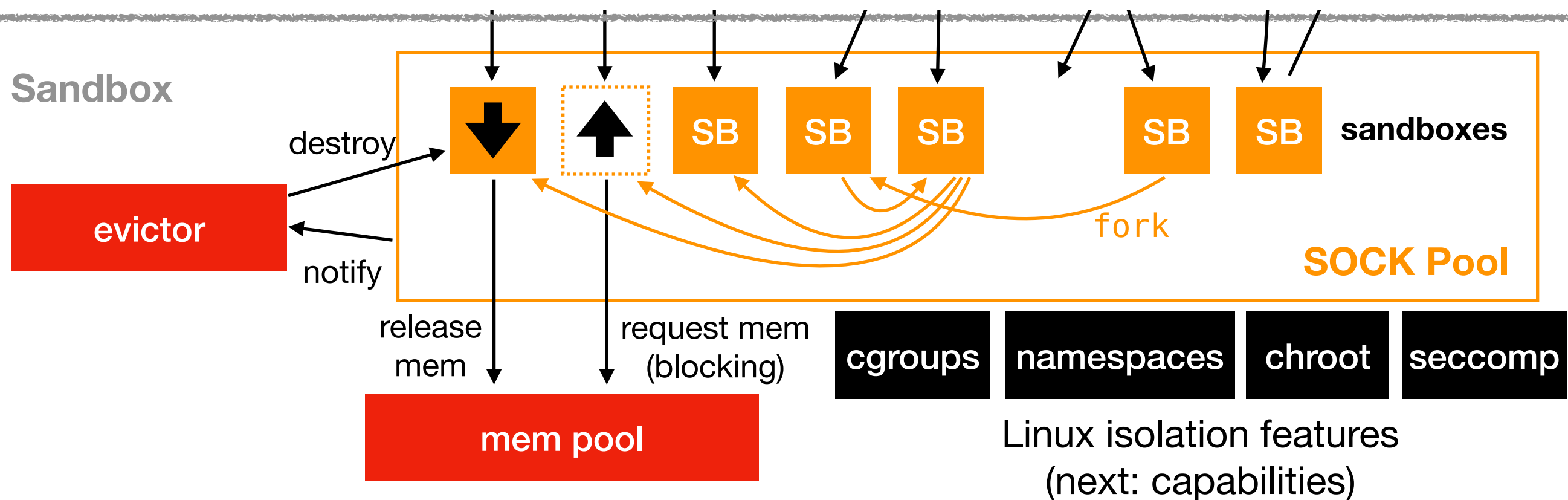
forward web requests

are Lambda processes
schedulable by Linux?

Sandbox

destroy

evictor

notify

release
mem

request mem
(blocking)

SB SB SB   SB SB **sandboxes**

fork

SOCK Pool

mem pool

cgroups   namespaces   chroot   seccomp

Linux isolation features
(next: capabilities)
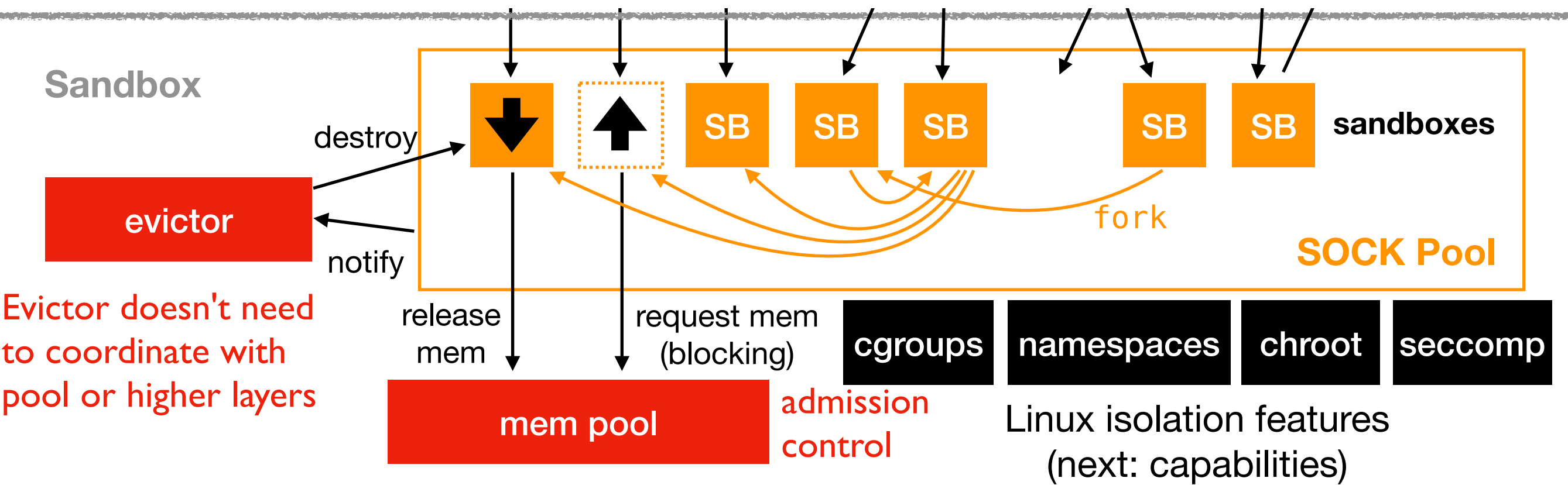
```go
type safeSandbox struct {
        Sandbox

        sync.Mutex
        paused        bool
        dead          error
        eventHandlers []SandboxEventFunc
}
```

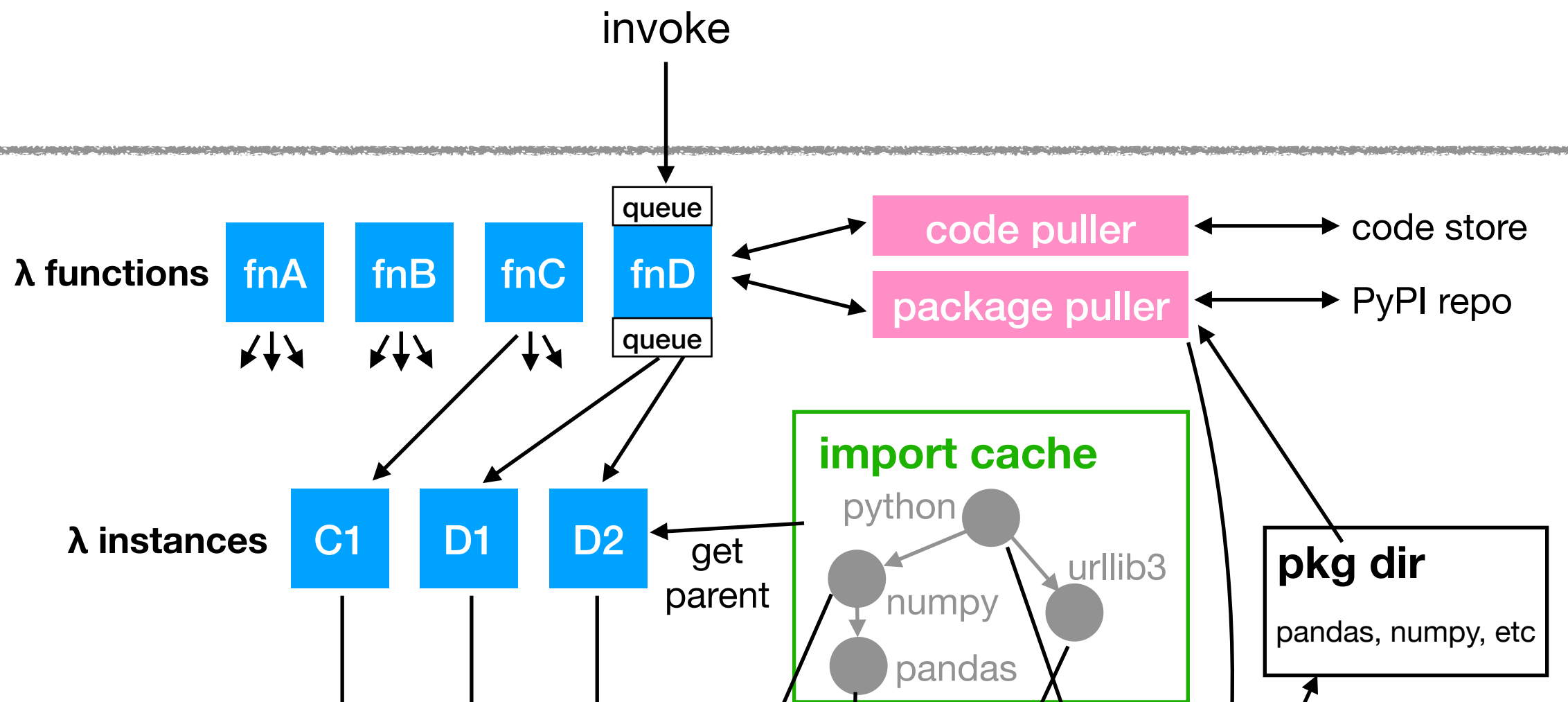safeSandbox is a Sandbox and wraps every other Sandbox implementation
- Sandbox API is threadsafe
- If any call returns any error, Sandbox is automatically destroyed (caller can just stop using it)
- Any call on destroyed Sandbox returns error, causes no harm
- notifies listeners (particularly evictor) that can operate independently
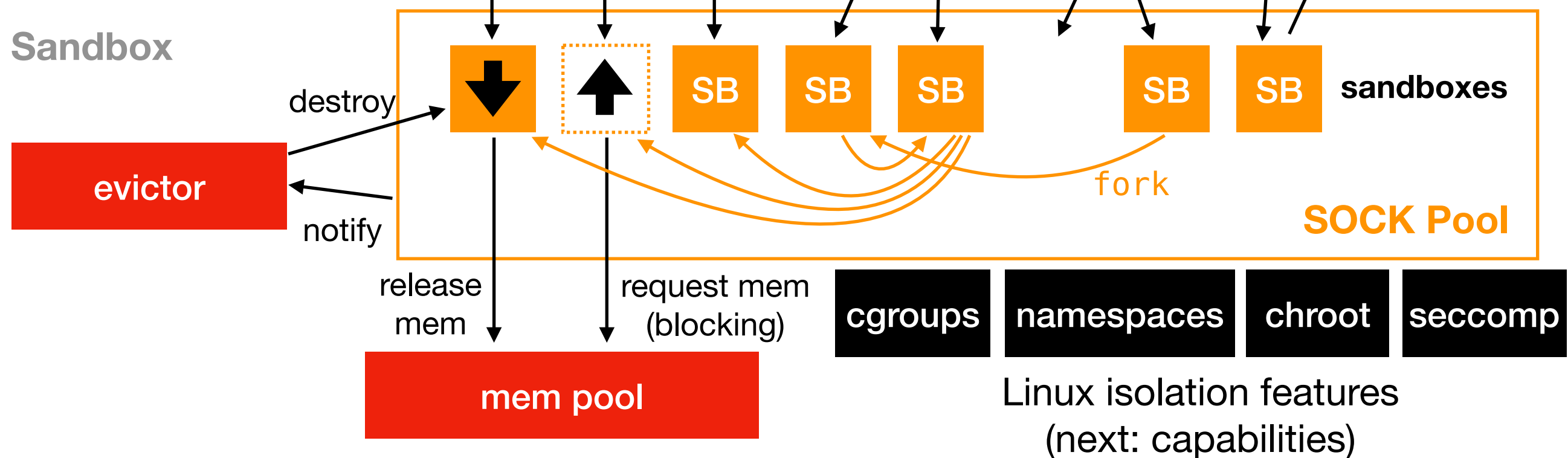
Sandbox

destroy

notify

SB SB SB SB SB **sandboxes**

fork

SOCK Pool

evictor

Evictor doesn't need to coordinate with pool or higher layers

release mem

request mem (blocking)

mem pool

admission control

cgroups   namespaces   chroot   seccomp

Linux isolation features (next: capabilities)

Event

invoke

Lambda

queue

code puller → code store

package puller ← → PyPI repo

Handles events; responsibilities:
1. fetch handler code (check for updates)
2. fetch package dependencies
3. allocate Sandboxes to run handlers

import cache

python

urllib3

numpy

pandas

λ instances   C1   D1   D2   ← get parent

pkg dir

pandas, numpy, etc

Sandbox

dest

evictor

noti

One sandbox pool used by three subsystems:
(1) Lambda instances, (2) import cache, (3)
package puller. Previous instances and cache
used different pools, and package puller was
(unsafely) not sandboxed.

SB   SB   **sandboxes**

ork

SOCK Pool

release mem → request mem (blocking) →

cgroups   namespaces   chroot   seccomp

mem pool

Linux isolation features
(next: capabilities)

Event

package puller uses Sandoxes to install packages locally because pip install may execute malicious code. OpenLambda now recursively discovers dependencies and support multiple versions of the same pkg.

Lambda

λ functions

fnA fnB fnC fnD

queue

code puller

code store

package puller ⟷ PyPI repo

λ instances

C1 D1 D2

get parent

import cache

python
numpy
pandas
urllib3

pkg dir

pandas, numpy, etc

Sandbox

destroy

SB SB SB SB SB **sandboxes**

fork

SOCK Pool

evictor

notify

release mem

request mem (blocking)

mem pool

cgroups namespaces chroot seccomp
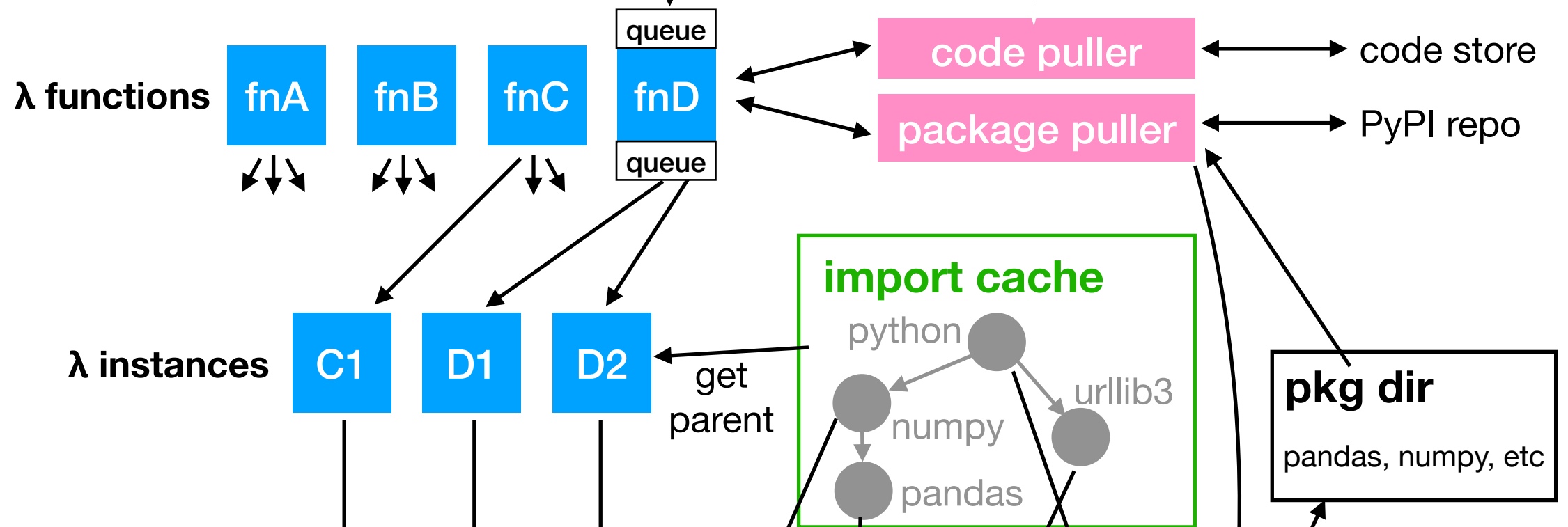
Linux isolation features
(next: capabilities)
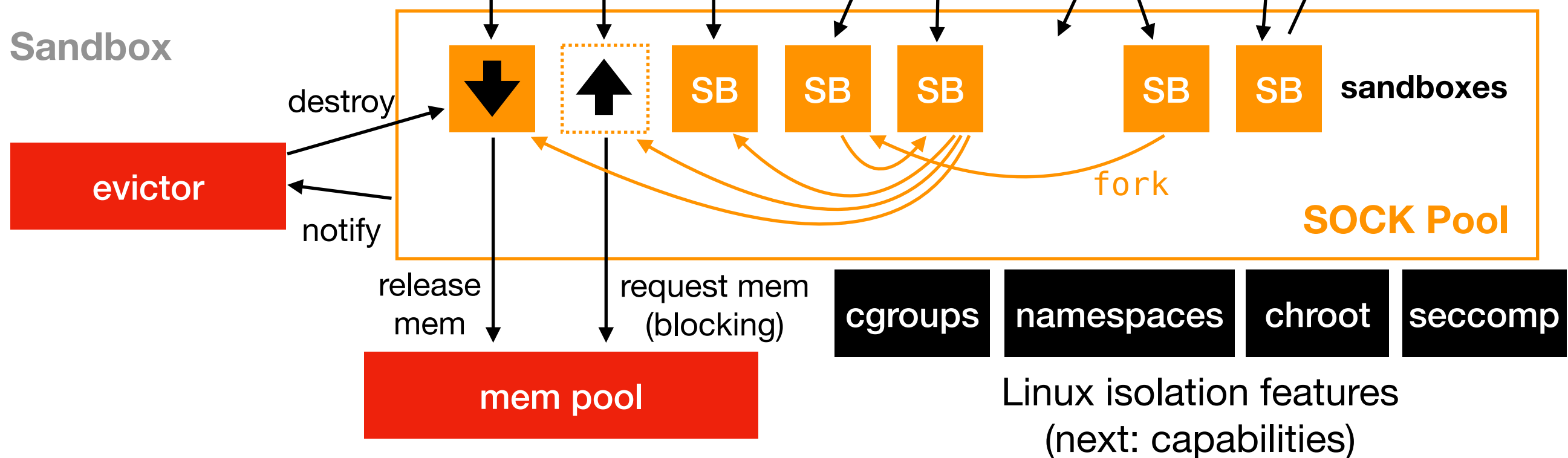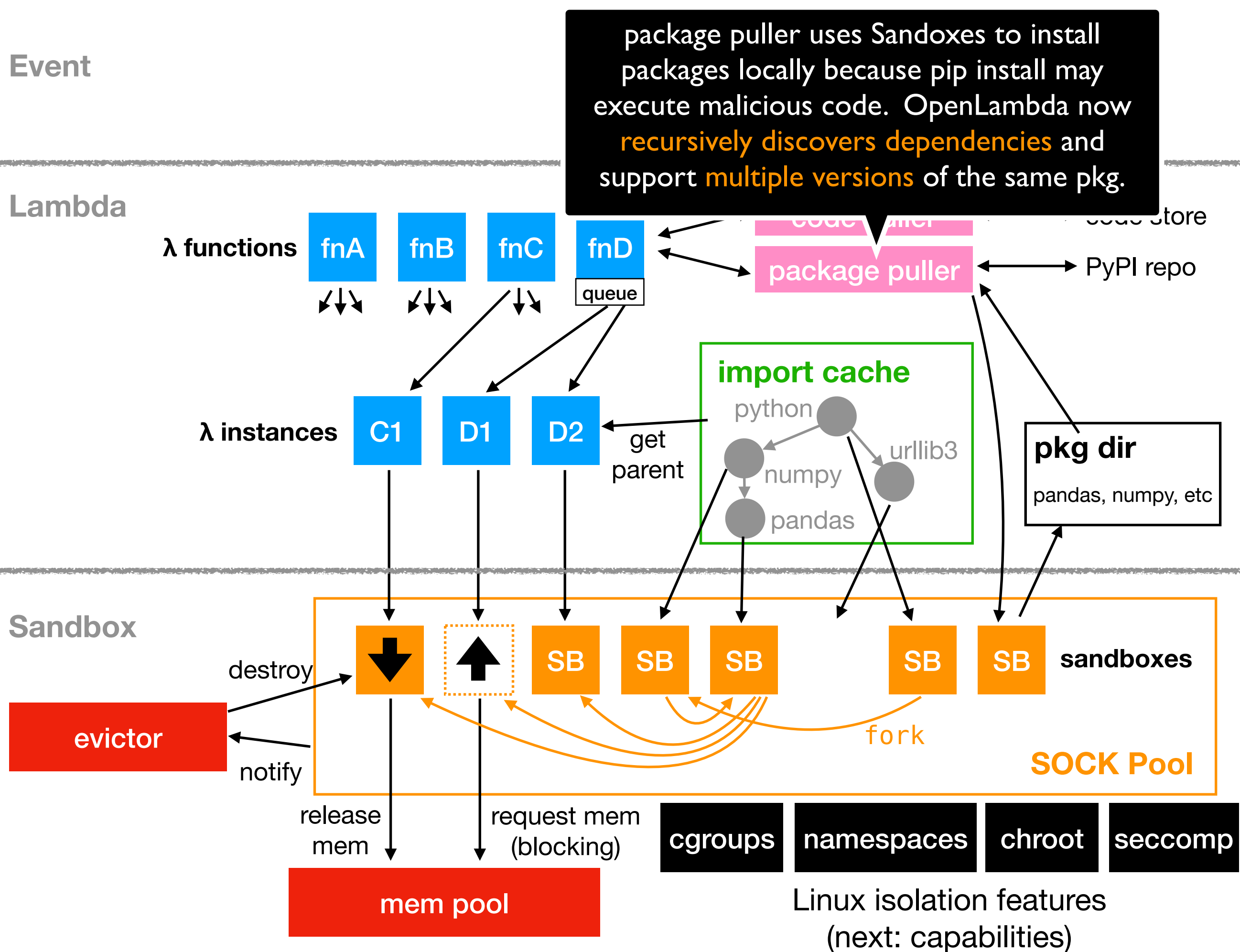
Event

invoke

- pulls code, packages
- kill stale instances if code changes
- decides number of instances based on queue depth, avg processing time
- returns `HTTP 429 Too Many` if queue is full

Lambda

λ functions

fnA fnB fnC fnD

queue
queue

import cache

λ instances

C1 D1 D2

Basically a robust, virtual Sandbox
- *sometimes* backed by physical SB
- masks Sandbox failure by restarting them

tc

Sandbox

destroy

SB SB SB SB SB sandboxes

evictor

notify

fork

SOCK Pool

release mem

request mem (blocking)

mem pool

cgroups namespaces chroot seccomp

Linux isolation features
(next: capabilities)

Event

Lambda

λ functions

fnA fnB fnC

queue

Lambda instances request parent zygotes from import cache to fork instance sandboxes
- zygote tree specified at startup
- each node not initially backed by actual Sandbox
- zygote created/evicted based on workload
- child node SBs forked from parent node SBs

package puller

import cache

python
urllib3
numpy
pandas

λ instances

C1 D1 D2

get parent

pkg dir

pandas, numpy, etc

Sandbox

SB SB SB SB SB sandboxes

destroy

evictor

notify

fork

SOCK Pool

release mem

request mem (blocking)

mem pool

cgroups namespaces chroot seccomp

Linux isolation features
(next: capabilities)