

# Rootless Containers in Gitpod

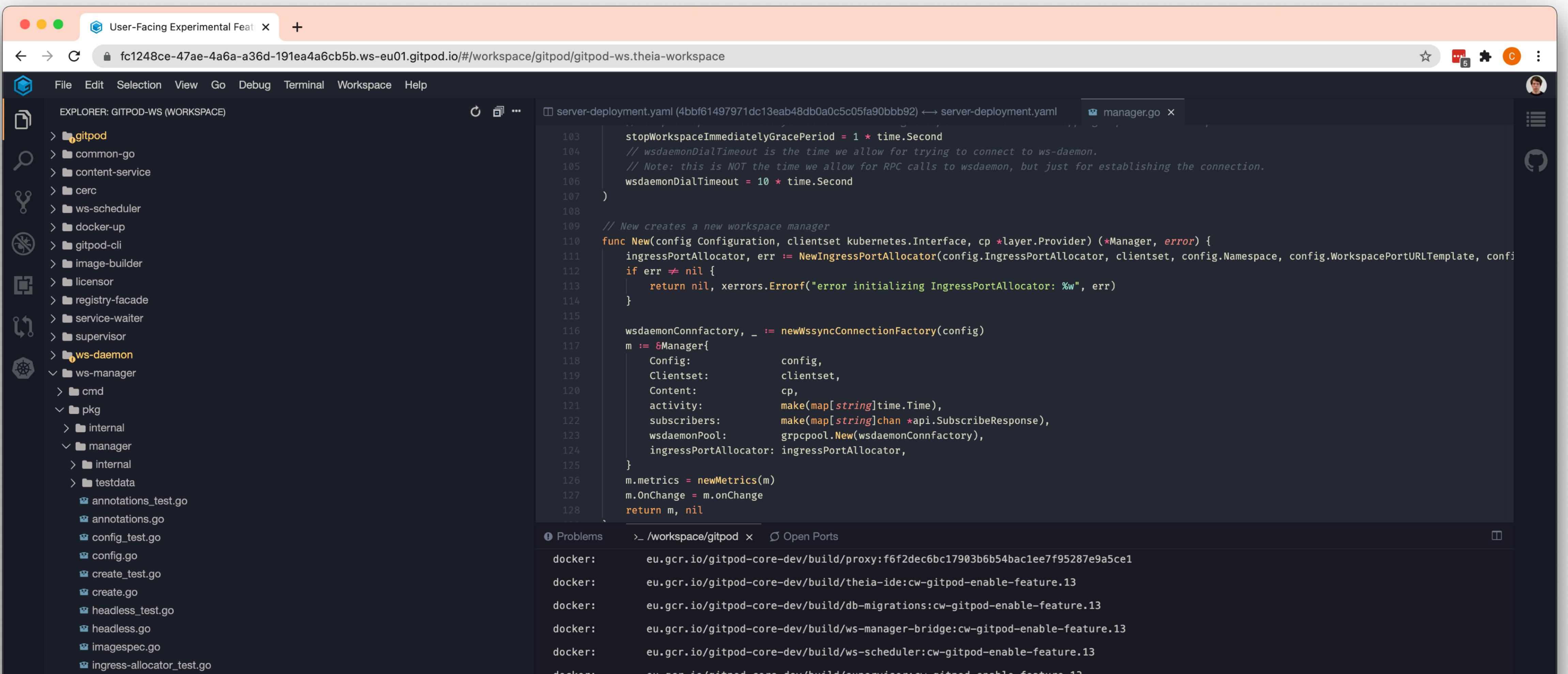
CNCF Webinar

Christian Weichel, PhD  
 @csweichel

Alban Crequy  
 @albcr

# Gitpod provides

automated dev environments



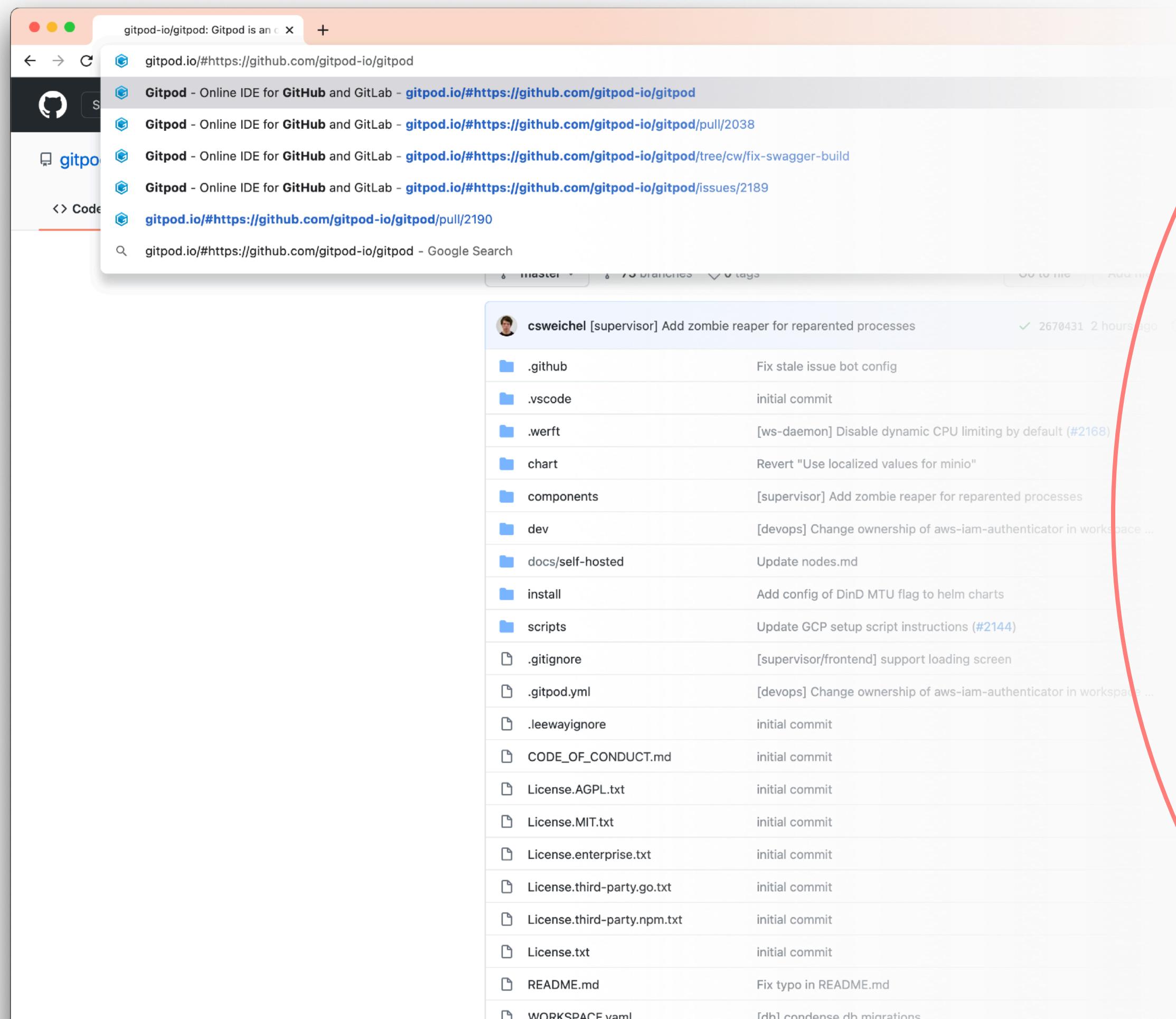
The screenshot shows a Gitpod workspace interface. The top bar displays the title "User-Facing Experimental Feat" and the URL "fc1248ce-47ae-4a6a-a36d-191ea4a6cb5b.ws-eu01.gitpod.io/#/workspace/gitpod/gitpod-ws.theia-workspace". The left sidebar contains a file tree under "EXPLORER: GITPOD-WS (WORKSPACE)" with several folders like "gitpod", "common-go", "content-service", "cerc", "ws-scheduler", "docker-up", "gitpod-cli", "image-builder", "licensor", "registry-facade", "service-waiter", "supervisor", "ws-daemon", "ws-manager", "cmd", "pkg", "internal", "manager", "internal", "testdata", and test files such as "annotations\_test.go", "config\_test.go", "create\_test.go", "create.go", "headless\_test.go", "headless.go", "imagespec.go", and "ingress-allocator\_test.go". The main area shows two tabs: "server-deployment.yaml" and "manager.go". The "manager.go" tab displays the following Go code:

```
103     stopWorkspaceImmediatelyGracePeriod = 1 * time.Second
104     // wsdaemonDialTimeout is the time we allow for trying to connect to ws-daemon.
105     // Note: this is NOT the time we allow for RPC calls to wsdaemon, but just for establishing the connection.
106     wsdaemonDialTimeout = 10 * time.Second
107
108     // New creates a new workspace manager
109     func New(config Configuration, clientset kubernetes.Interface, cp *layer.Provider) (*Manager, error) {
110         ingressPortAllocator, err := NewIngressPortAllocator(config.IngressPortAllocator, clientset, config.Namespace, config.WorkspacePortURLTemplate, config)
111         if err != nil {
112             return nil, xerrors.Errorf("error initializing IngressPortAllocator: %w", err)
113         }
114
115         wsdaemonConnfactory, _ := newWssyncConnectionFactory(config)
116         m := &Manager{
117             Config:           config,
118             Clientset:        clientset,
119             Content:          cp,
120             Activity:         make(map[string]time.Time),
121             Subscribers:      make(map[string]chan *api.SubscribeResponse),
122             WsdaemonPool:     grpcpool.New(wsdaemonConnfactory),
123             IngressPortAllocator: ingressPortAllocator,
124         }
125         m.metrics = newMetrics(m)
126         m.OnChange = m.onChange
127         return m, nil
128     }
```

At the bottom, there are "Problems" and "Open Ports" sections. The "Problems" section lists several Docker images from "eu.gcr.io": "gitpod-core-dev/build/proxy:f6f2dec6bc17903b6b54bac1ee7f95287e9a5ce1", "gitpod-core-dev/build/theia-ide:cw-gitpod-enable-feature.13", "gitpod-core-dev/build/db-migrations:cw-gitpod-enable-feature.13", "gitpod-core-dev/build/ws-manager-bridge:cw-gitpod-enable-feature.13", "gitpod-core-dev/build/ws-scheduler:cw-gitpod-enable-feature.13", and "gitpod-core-dev/build/ws-scheduler:cw-gitpod-enable-feature.13". The "Open Ports" section shows ports mapped to these Docker images.

# Gitpod provides

# automated dev environments in Kubernetes



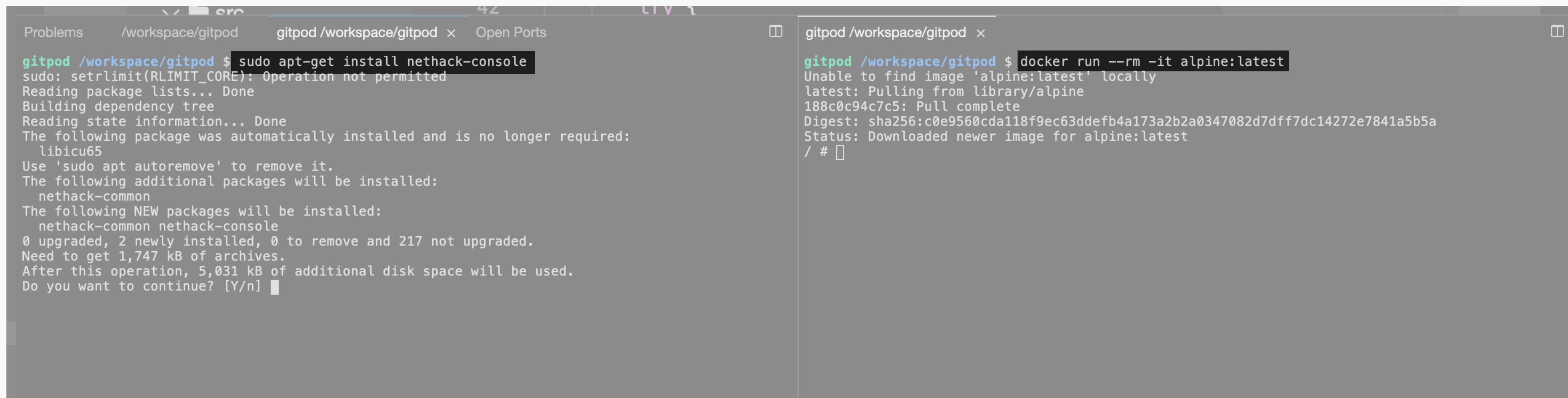
# How it started

no sudo, no apt-get install, no Docker

```
gitpod /workspace/gitpod $ sudo apt-get install docker.io
sudo: effective uid is not 0, is /usr/bin/sudo on a file system with the 'nosuid' option set or an NFS file system without root privileges?
gitpod /workspace/gitpod $ docker run --rm -it alpine:latest
docker: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?.
See 'docker run --help'.
gitpod /workspace/gitpod $ █
```

# How it's going

sudo, apt-get install, Docker



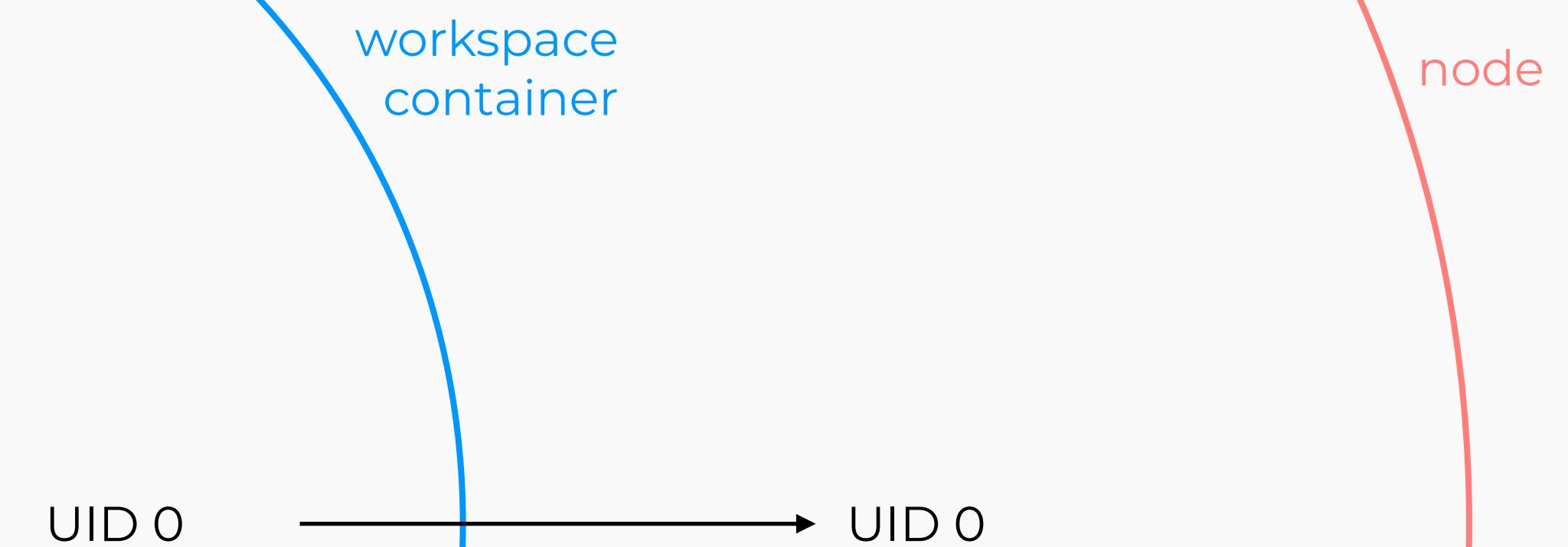
```
gitpod /workspace/gitpod $ sudo apt-get install nethack-console
sudo: setrlimit(RLIMIT_CORE): Operation not permitted
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libicu65
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  nethack-common
The following NEW packages will be installed:
  nethack-common nethack-console
0 upgraded, 2 newly installed, 0 to remove and 217 not upgraded.
Need to get 1,747 kB of archives.
After this operation, 5,031 kB of additional disk space will be used.
Do you want to continue? [Y/n] 
```

```
gitpod /workspace/gitpod $ docker run --rm -it alpine:latest
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
188c0c94c7c5: Pull complete
Digest: sha256:c0e9560cda118f9ec63ddefb4a173a2b2a0347082d7dff7dc14272e7841a5b5a
Status: Downloaded newer image for alpine:latest
/ # 
```

# Naïve: give privileges on the node

Don't do this at home

```
● ● ●  
  
spec:  
  allowPrivilegeEscalation: true  
  allowedCapabilities:  
    - '*'  
  fsGroup:  
    rule: RunAsAny  
  hostIPC: true  
  hostNetwork: true  
  hostPID: true  
  hostPorts:  
    - max: 65535  
      min: 0  
  privileged: true  
  runAsUser:  
    rule: RunAsAny  
  seLinux:  
    rule: RunAsAny  
  supplementalGroups:  
    rule: RunAsAny  
  volumes:  
    - '*'
```



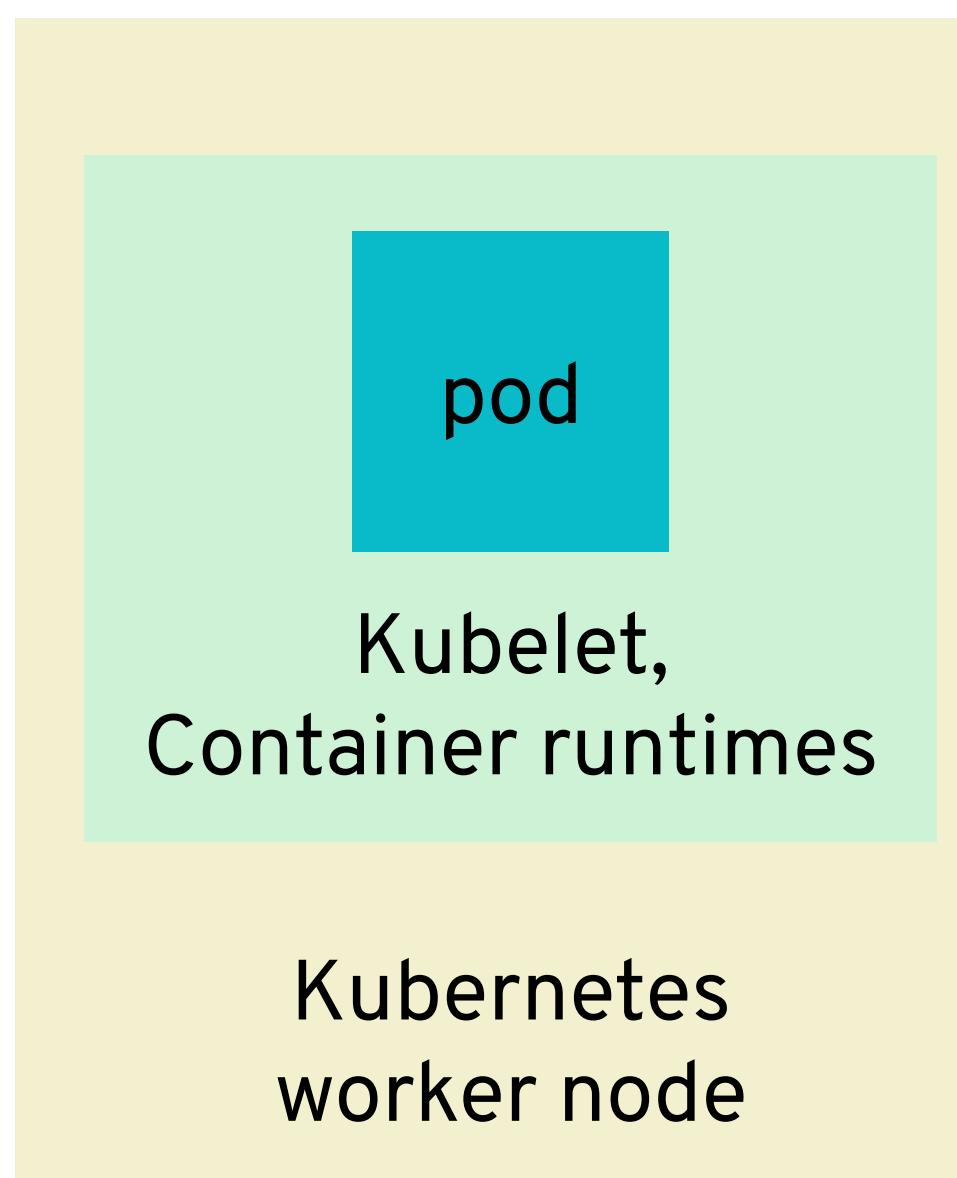
# Overview of isolation tech (1)

- VM-like Container Runtimes
  - Nabla containers: unikernel built specifically for the workload
  - gVisor: application kernel, written in Go, that implements a substantial portion of the Linux system surface
  - Kata Containers: container runtime, building lightweight virtual machines
  - Firecracker: Virtual Machine Manager like QEMU
- Limitations
  - Compatibility
  - Network or filesystem performance
  - Density

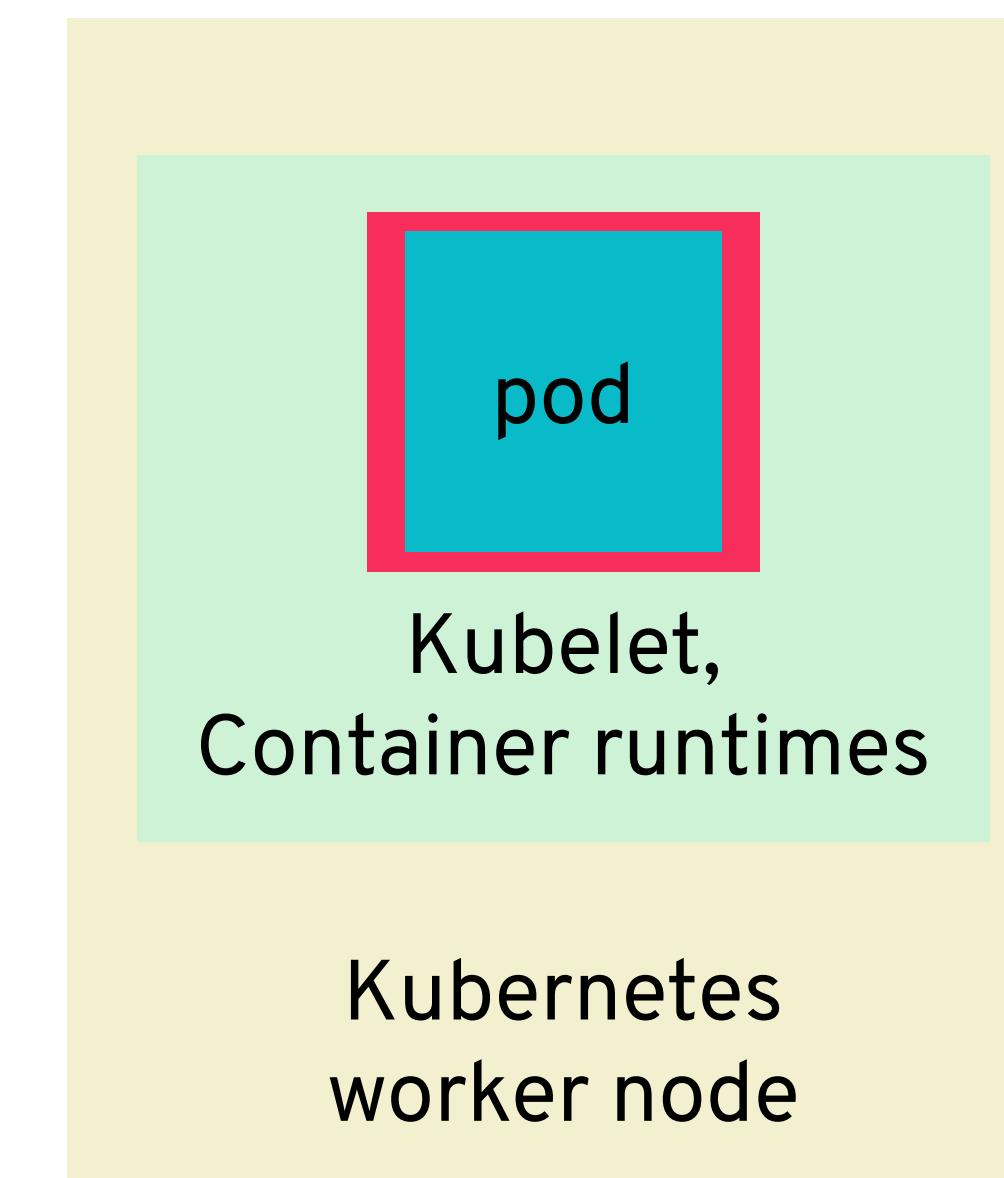


# User namespaces in Kubernetes

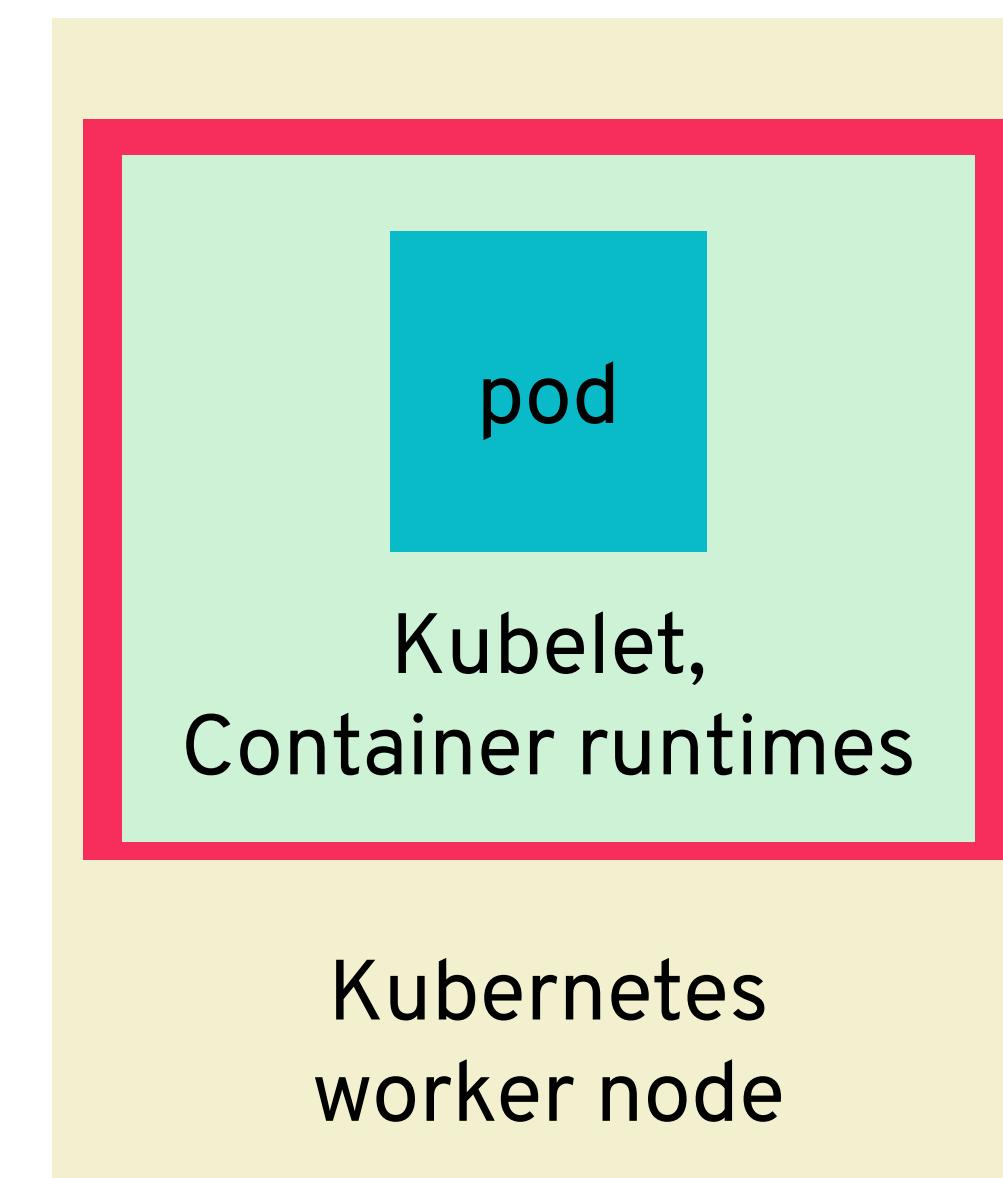
Three different ways to use user namespaces in Kubernetes



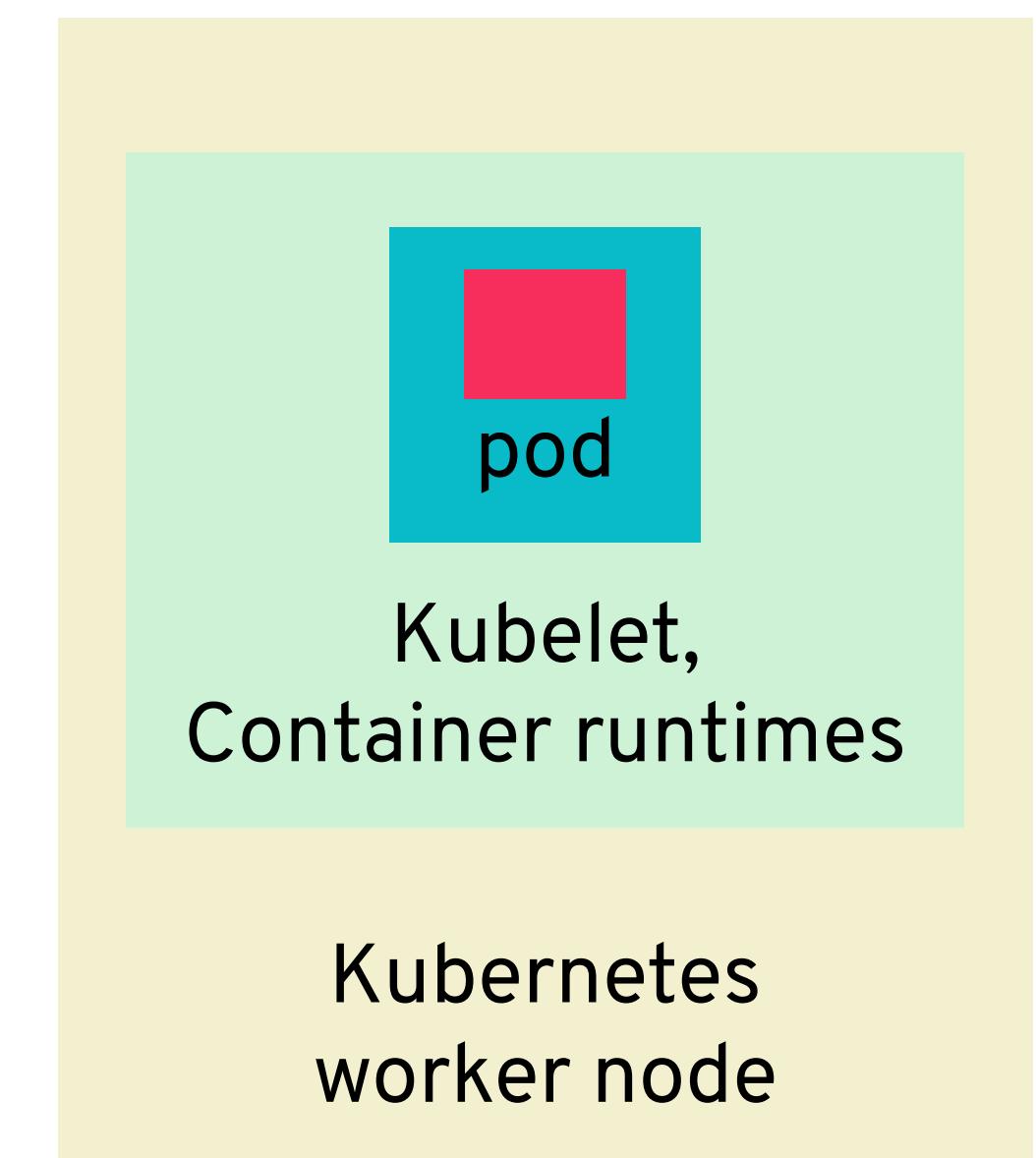
Without user namespaces



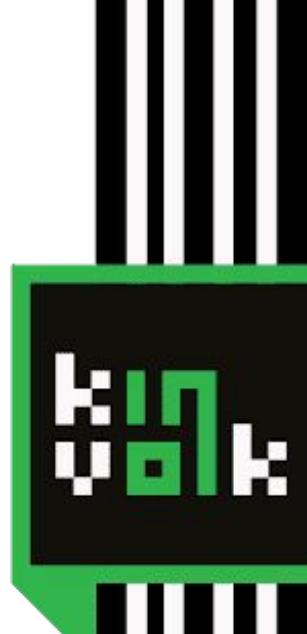
KEP-127:  
New field in PodSpec  
hostUserNamespace=



KEP-2033:  
rootless mode



Workload in the pod  
creating  
user namespaces  
(Gitpod)



# User Namespaces

a rough sketch

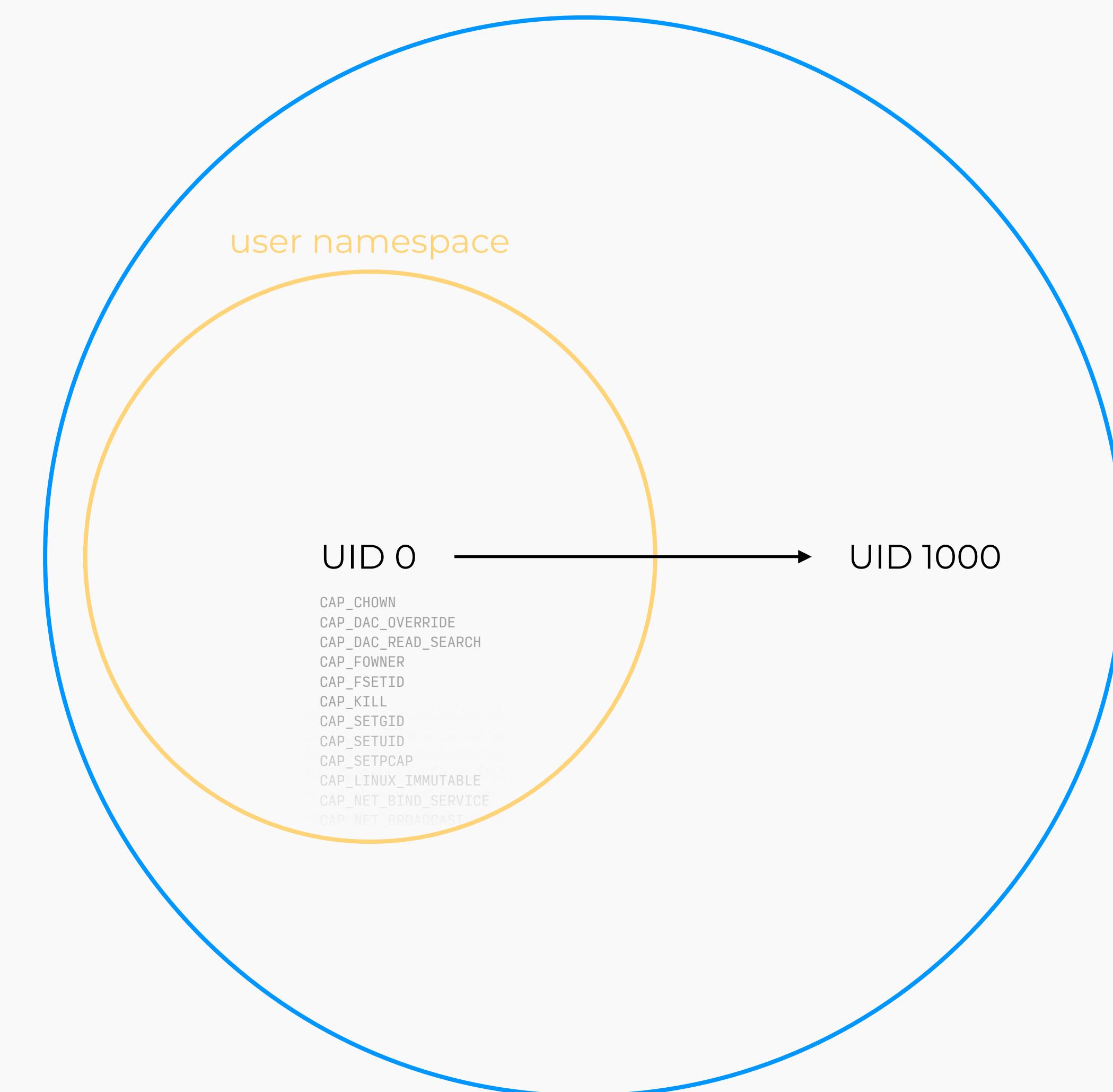
unshare()

⚡ write(/proc/<pid>/uid\_map)  
⚡ write(/proc/<pid>/gid\_map)

execve()



```
$ strace -f -e openat,write,unshare unshare -U -r echo
```



# Challenge I

writing `/proc/.../{u,g}id_map`

Gitpod  
ws-daemon

- ⚡ hostPID := translatePID(pid: 2)
- 🚀 write(`/proc/<hostPID>/uid_map`)
- 🚀 write(`/proc/<hostPID>/gid_map`)

`writeMapping(pid: 2)`

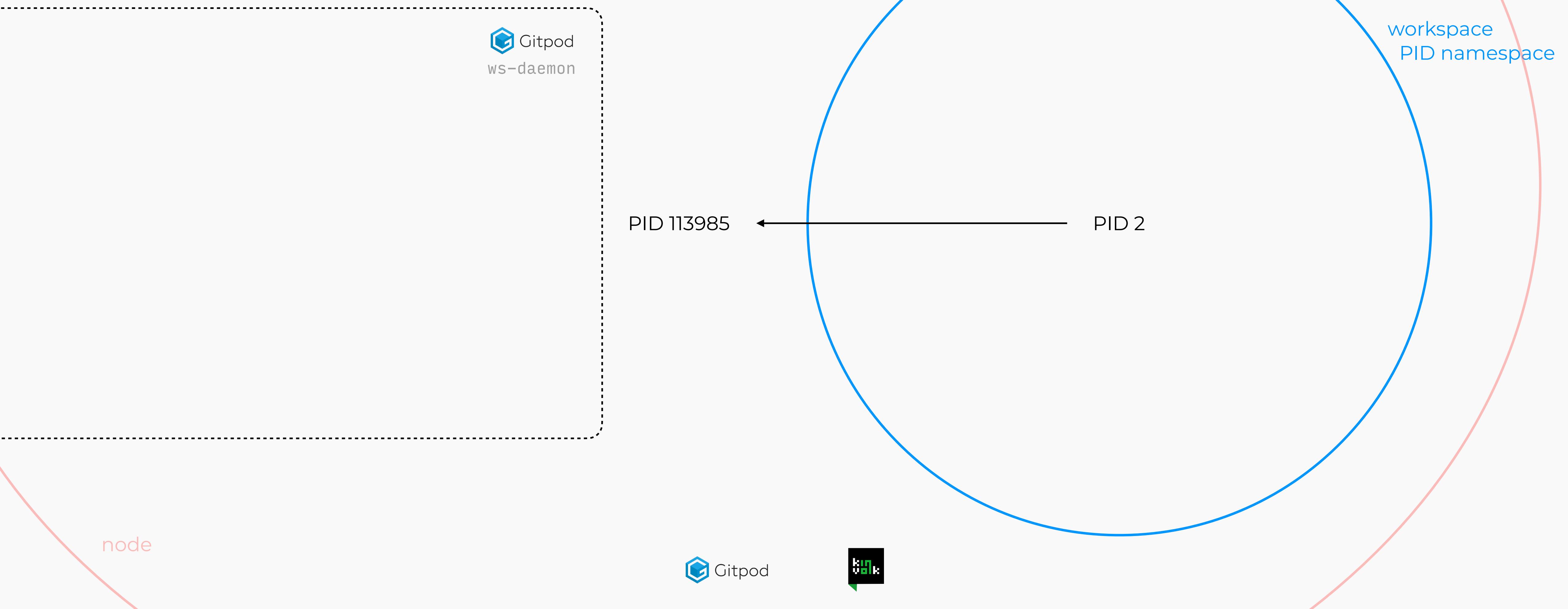


workspace  
container

node

# Challenge II

PID mapping



# Challenge II

## PID mapping

```
root@ws-daemon-8nh7g:/# cat /proc/2548604/status
Name: sleep
Umask: 0022
State: S (sleeping)
Tgid: 2548604
Ngid: 0
Pid: 2548604
PPid: 27259
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 256
Groups: 1 2 3 4 6 10 11 20 26 27
NSTgid: 2548604 1349
NSpid: 2548604 1349
NSpgid: 27259 1
NSsid: 27259 1
VmPeak: 3520 kB
VmSize: 1492 kB
VmLck: 0 kB
VmPin: 0 kB
VmHWM: 4 kB
VmRSS: 4 kB
RssAnon: 4 kB
RssFile: 0 kB
RssShmem: 0 kB
VmData: 20 kB
VmStk: 132 kB
VmExe: 756 kB
VmLib: 552 kB
VmPTE: 48 kB
VmSwap: 0 kB
HugetlbPages: 0 kB
CoreDumping: 0
THP_enabled: 1
Threads: 1
SigQ: 3/120489
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000000
SigCgt: 0000000000000000
CapInh: 0000003fffffff
CapPrm: 0000003fffffff
CapEff: 0000003fffffff
CapBnd: 0000003fffffff
CapAmb: 0000000000000000
NoNewPrivs: 0
Seccomp: 0
```

“container process”

“target process”

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2970734	root	20	0	1020	4	0	S	0.0	0.0	0:00.43	`- /pause
2972768	root	20	0	10796	4912	4340	S	0.0	0.0	0:00.07	`- containerd-shim -namespace k8s.io -workdir
2972788	33333	20	0	/16576	15752	11228	S	0.0	0.1	0:00.45	`- ./supervisor/supervisor ring0
2972959	33333	30	10	717984	15312	10716	S	0.0	0.0	0:00.07	`- /proc/self/exe ring1 --mapping-esta
2973008	33333	30	10	718240	14860	10332	S	0.0	0.0	0:00.05	`- /proc/self/exe ring2
2973027	133332	30	10	720096	22856	12888	S	0.3	0.1	0:00.63	`- /proc/self/exe run --inns
2973115	133332	25	5	1400816	118844	31028	S	0.0	0.4	0:05.82	`- /theia/node/bin/gitpod-
2975354	133332	25	5	284184	43480	28728	S	0.3	0.1	0:00.78	`- /theia/node/bin/git
2975533	133332	26	6	12924	9848	3544	S	0.0	0.0	0:00.15	`- /bin/bash -i
2976781	133332	20	0	718756	16140	11100	S	0.0	0.1	0:00.08	`- ./supervisor/su
2976358	133332	25	5	297348	61372	29576	S	0.0	0.2	0:00.91	`- /theia/node/bin/git
2976807	133332	25	5	238588	39580	28460	S	0.0	0.1	0:00.19	`- /theia/node/bin
2973194	133332	26	6	12780	9852	3608	S	0.0	0.0	0:00.18	`- /bin/bash -i -l
2977954	133332	30	10	5488	604	528	S	0.0	0.0	0:00.00	`- sleep 1234



# Challenge II

PID mapping

Gitpod  
ws-daemon

- hostPID := translatePID(pid: 2)
- write(/proc/<hostPID>/uid\_map)
- write(/proc/<hostPID>/gid\_map)

← writeMapping(pid: 2)



node

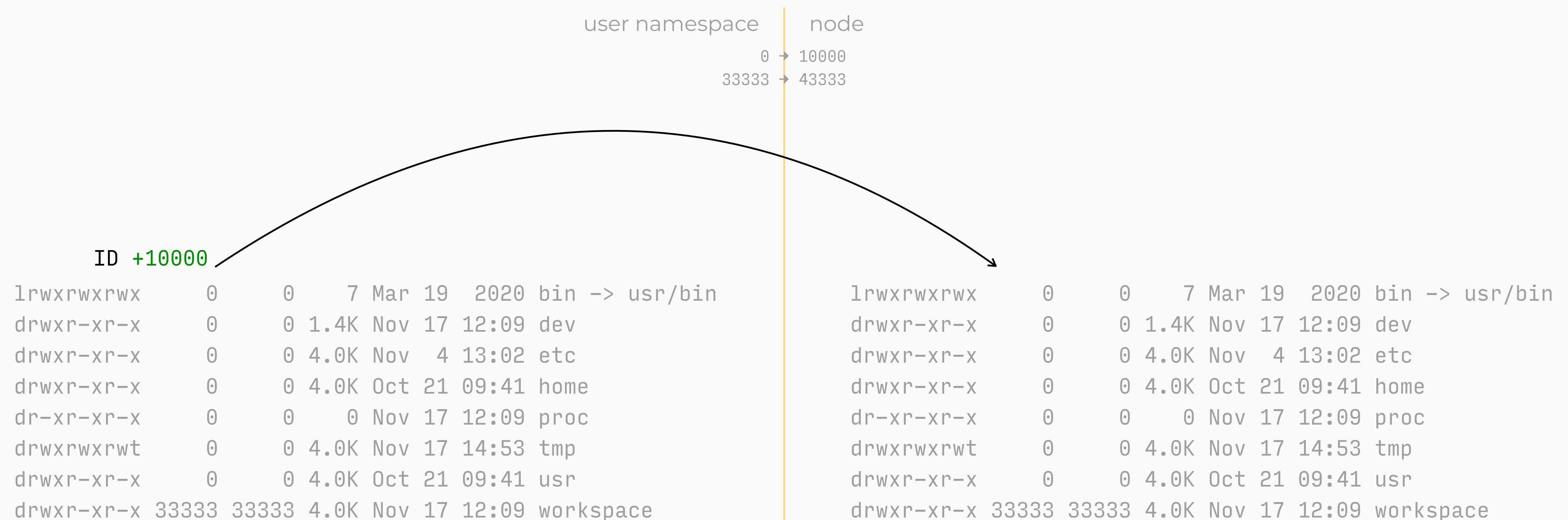
# Challenge III

filesystem UID/GID shift

```
gitpod / $ unshare -U -r
root / # ls -lhn /
total 72K
lrwxrwxrwx  1 65534 65534    7 Mar 19  2020 bin -> usr/bin
drwxr-xr-x  2 65534 65534 4.0K Mar  6  2020 boot
drwxr-xr-x 10 65534 65534 1.4K Nov 17 12:09 dev
drwxr-xr-x  1 65534 65534 4.0K Nov  4 13:02 etc
drwxr-xr-x  1 65534 65534 4.0K Oct 21 09:41 home
lrwxrwxrwx  1 65534 65534    7 Mar 19  2020 lib -> usr/lib
lrwxrwxrwx  1 65534 65534    9 Mar 19  2020 lib32 -> usr/lib32
lrwxrwxrwx  1 65534 65534    9 Mar 19  2020 lib64 -> usr/lib64
lrwxrwxrwx  1 65534 65534   10 Mar 19  2020 libx32 -> usr/libx32
drwxr-xr-x  2 65534 65534 4.0K Mar 19  2020 media
drwxr-xr-x  2 65534 65534 4.0K Mar 19  2020 mnt
drwxr-xr-x  1      0      0 4.0K Nov  4 12:52 opt
dr-xr-xr-x 674 65534 65534      0 Nov 17 12:09 proc
drwx-----  2 65534 65534 4.0K Mar 19  2020 root
drwxr-xr-x  1 65534 65534 4.0K Apr 15  2020 run
lrwxrwxrwx  1 65534 65534    8 Mar 19  2020 sbin -> usr/sbin
drwxr-xr-x  2 65534 65534 4.0K Mar 19  2020 srv
dr-xr-xr-x 12 65534 65534      0 Nov 17 14:52 sys
drwxr-xr-x  1 65534 65534 4.0K Jul 14 21:18 theia
drwxrwxrwt  1 65534 65534 4.0K Nov 17 14:53 tmp
drwxr-xr-x  1 65534 65534 4.0K Oct 21 09:41 usr
drwxr-xr-x  1 65534 65534 4.0K Nov  4 13:02 var
drwxr-xr-x  5      0      0 4.0K Nov 17 12:09 workspace
root / #
```

# Challenge III

filesystem UID/GID shift



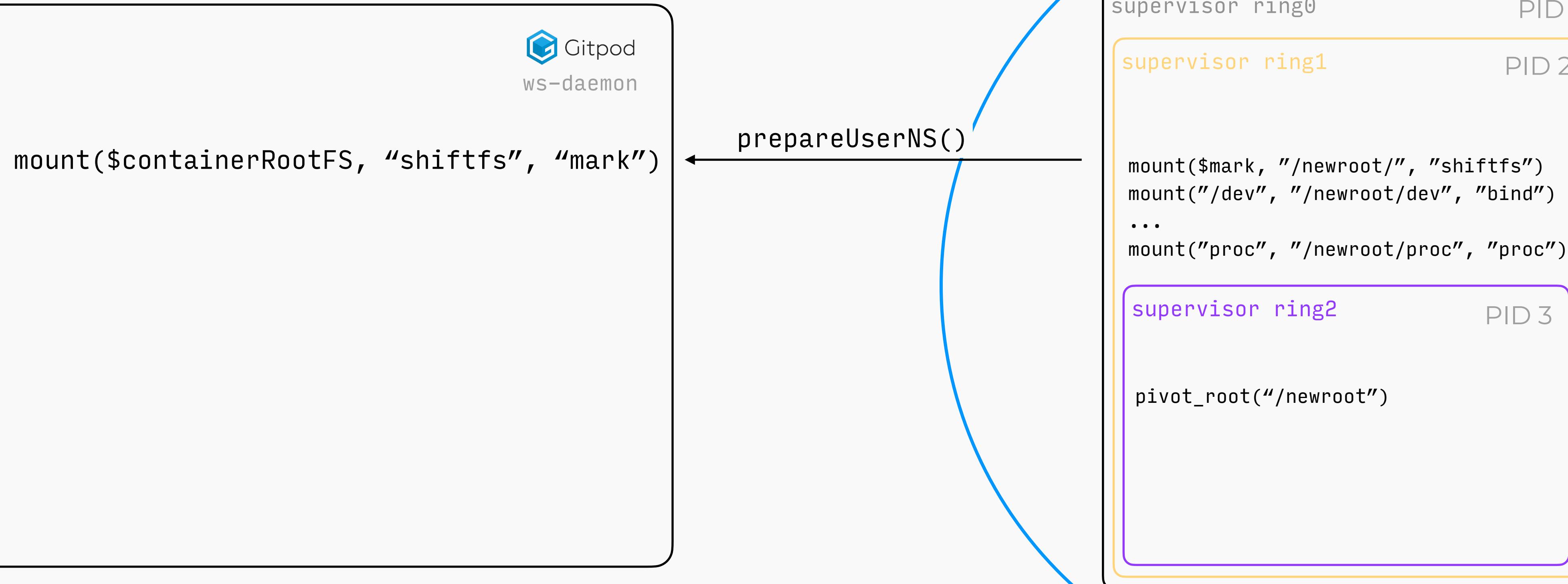
# Challenge III

filesystem UID/GID shift

	<b>fuse-overlayfs</b>	<b>overlayfs metacopy</b>	<b>shiftfs</b>
<b>Namespaced</b>	★★★ yes	★★ yes - on Ubuntu	★ requires privileged mark mount
<b>Upfront Cost</b>	★★★ none	★ very high - chown -R *	★★★ none
<b>Runtime Cost</b>	★ very high - userland process	★★ comparatively low	★★★ virtually none
<b>Platform Specific</b>	★★★ no	★★ works best on Ubuntu	★ Ubuntu only

# Challenge III

filesystem UID/GID shift

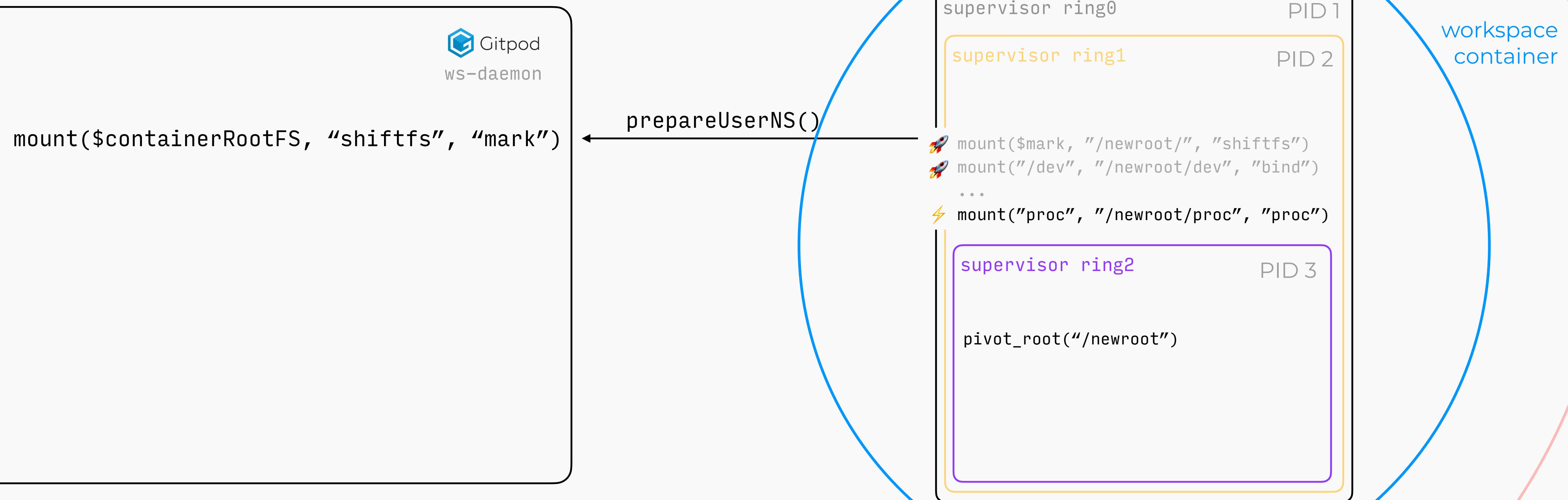


node

user namespace  
PID namespace

# Challenge IV

mounting /proc



node

# Challenge IV

mounting /proc

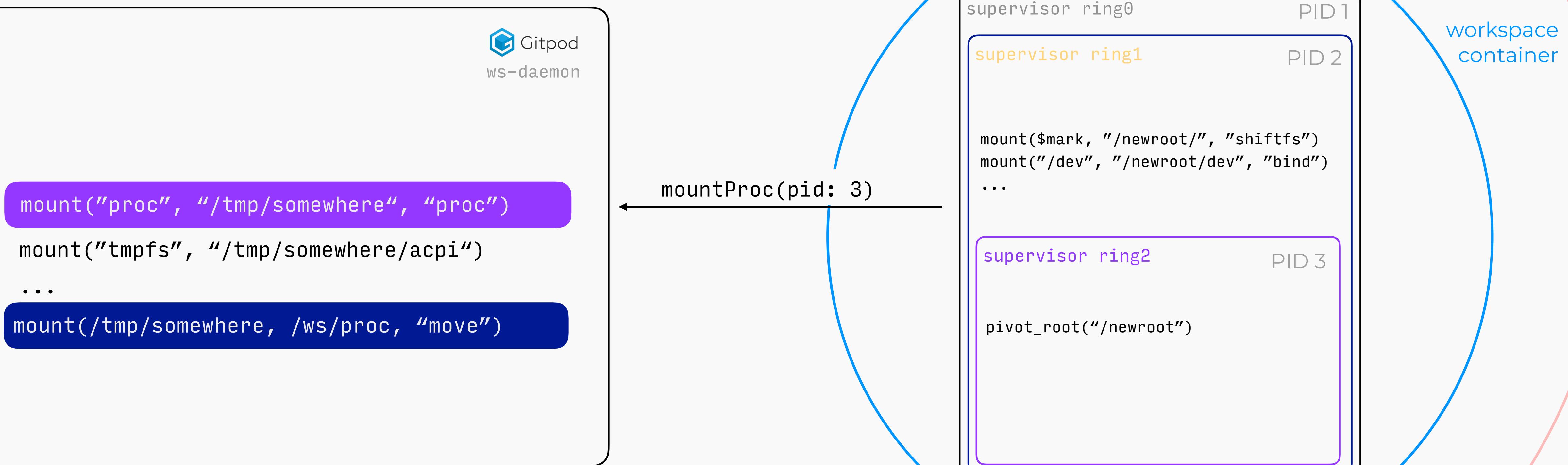


```
$ mount | grep proc
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
proc on /proc/bus type proc (ro,relatime)
proc on /proc/fs type proc (ro,relatime)
proc on /proc/irq type proc (ro,relatime)
proc on /proc/sys type proc (ro,relatime)
proc on /proc/sysrq-trigger type proc (ro,relatime)
tmpfs on /proc/acpi type tmpfs (ro,relatime)
tmpfs on /proc/kcore type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/keys type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/timer_list type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/sched_debug type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/scsi type tmpfs (ro,relatime)
```

# Challenge IV

filesystem UID/GID shift

- user namespace
- PID namespace
- mount namespace



# We got root - but no Docker yet



# Docker in rootless workspaces

Heaps of prior art



Akihiro Suda  
 @\_AkihiroSuda\_



[https://kinvolk.io/blog/2018/04/  
towards-unprivileged-container-builds/](https://kinvolk.io/blog/2018/04/towards-unprivileged-container-builds/)

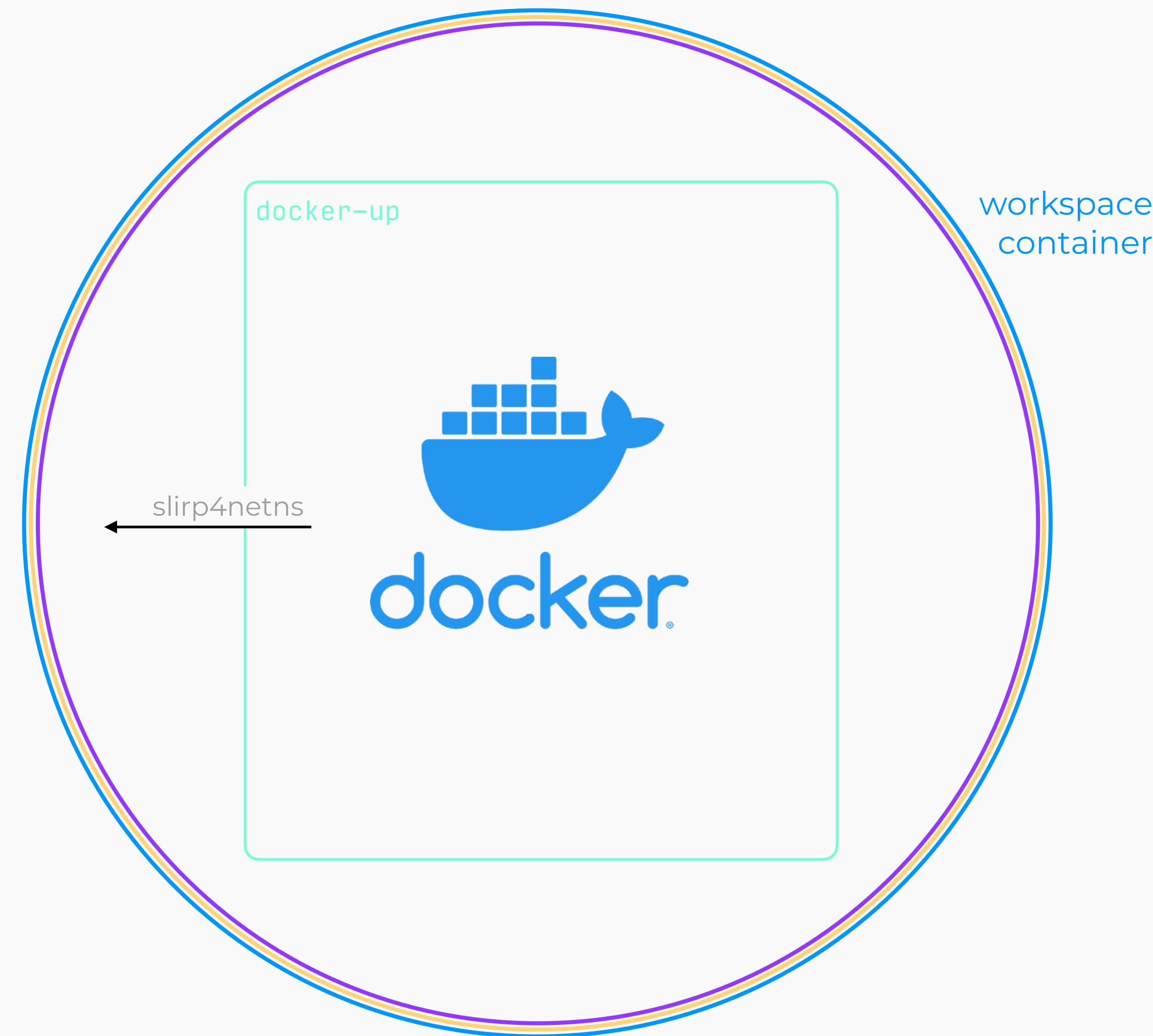
 Rootless Containers

<https://rootlesscontainers.rs/>

- █ user namespace
- █ PID namespace
- █ Network namespace

# Docker in rootless workspaces

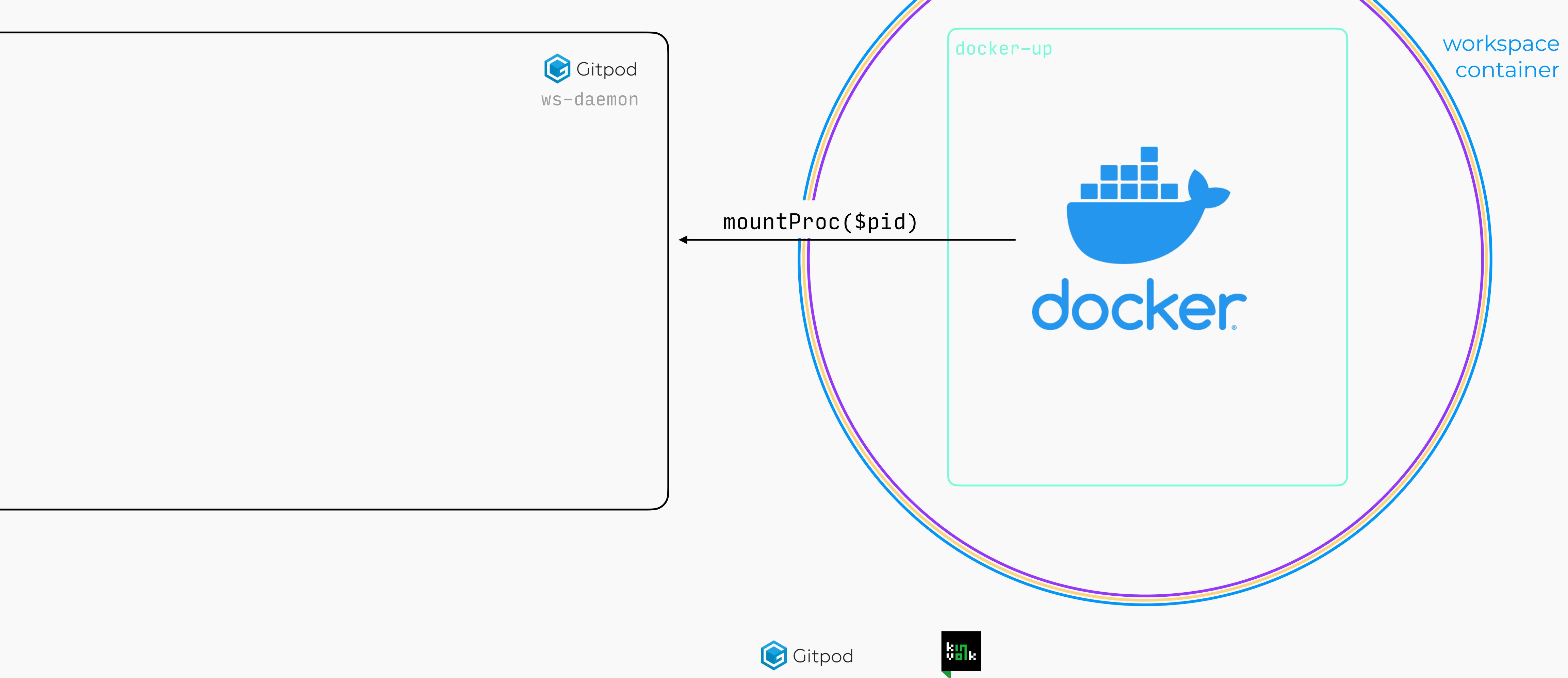
Networking



- █ user namespace
- █ PID namespace
- █ Network namespace

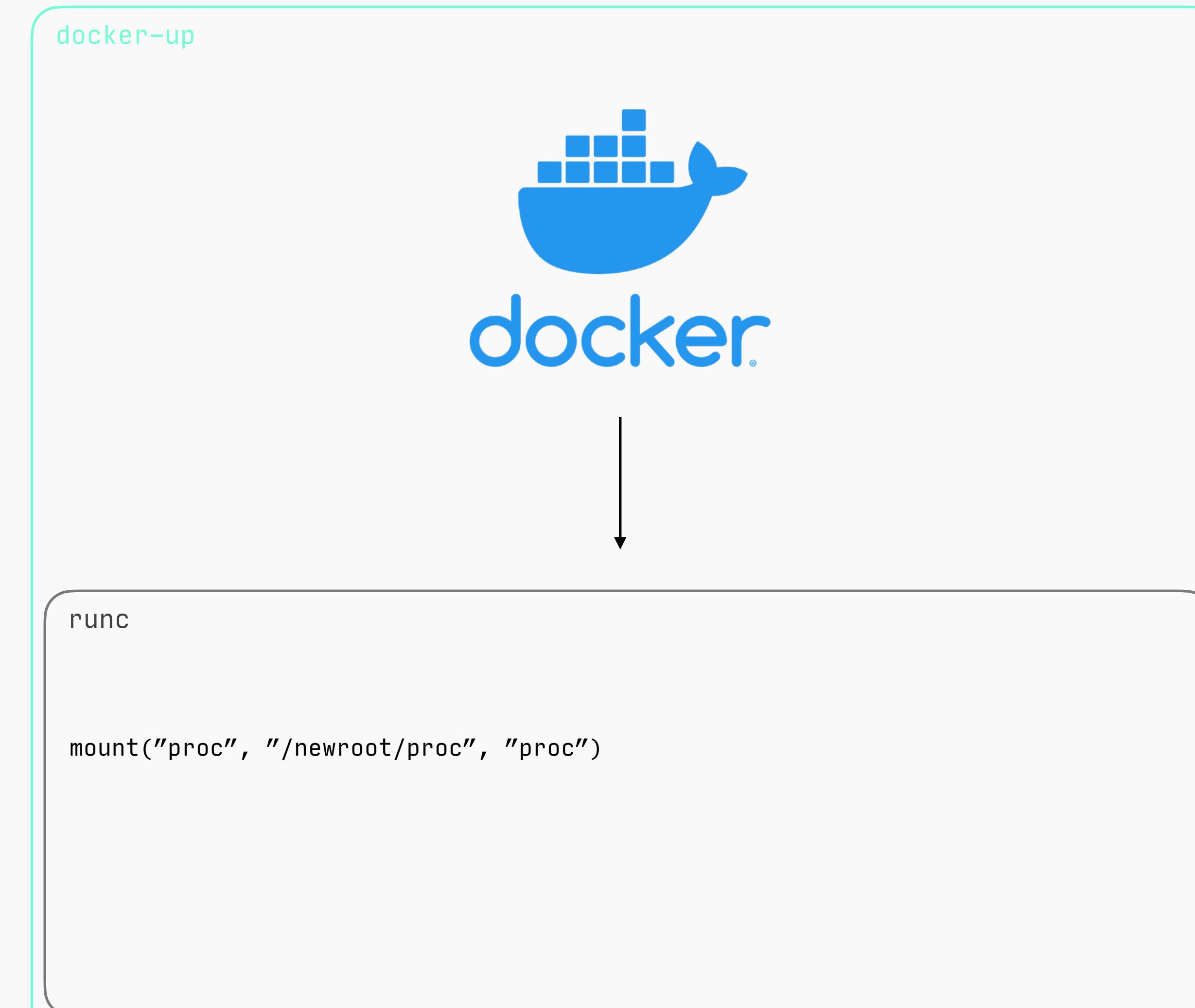
# Docker in rootless workspaces

mounting /proc



# Docker in rootless workspaces

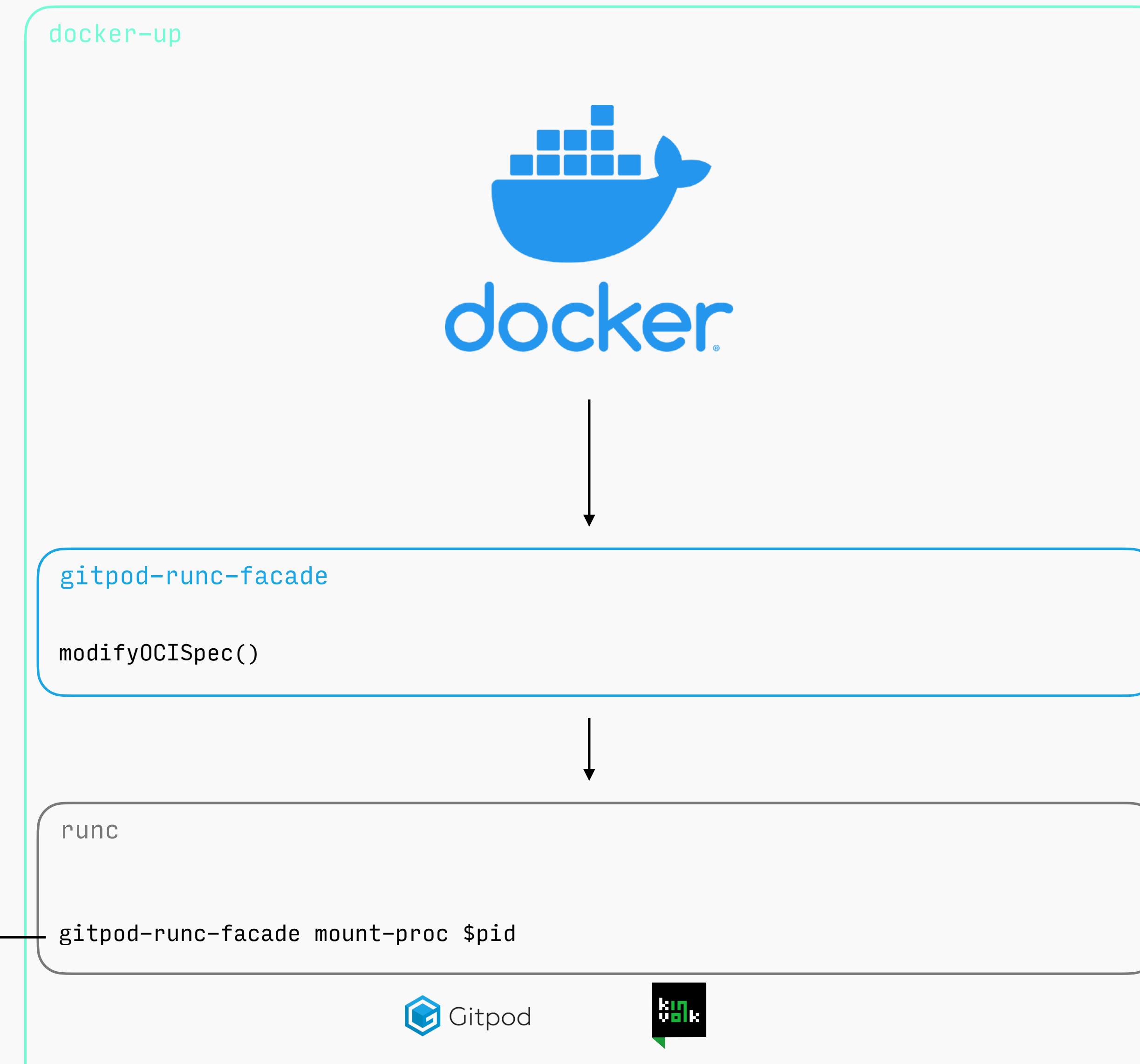
mounting /proc



# Docker in rootless workspaces

mounting /proc

- user namespace
- PID namespace
- Network namespace





# Idmapped mounts

- Currently, Gitpod uses shiftfs from a kernel module, not in Linux upstream
- New: ID-mapped mounts patch set
  - Attach idmappings to bind mounts
  - Shifting of a container rootfs or base image without having to mangle every file
  - Sharing of data between host with underprivileged containers

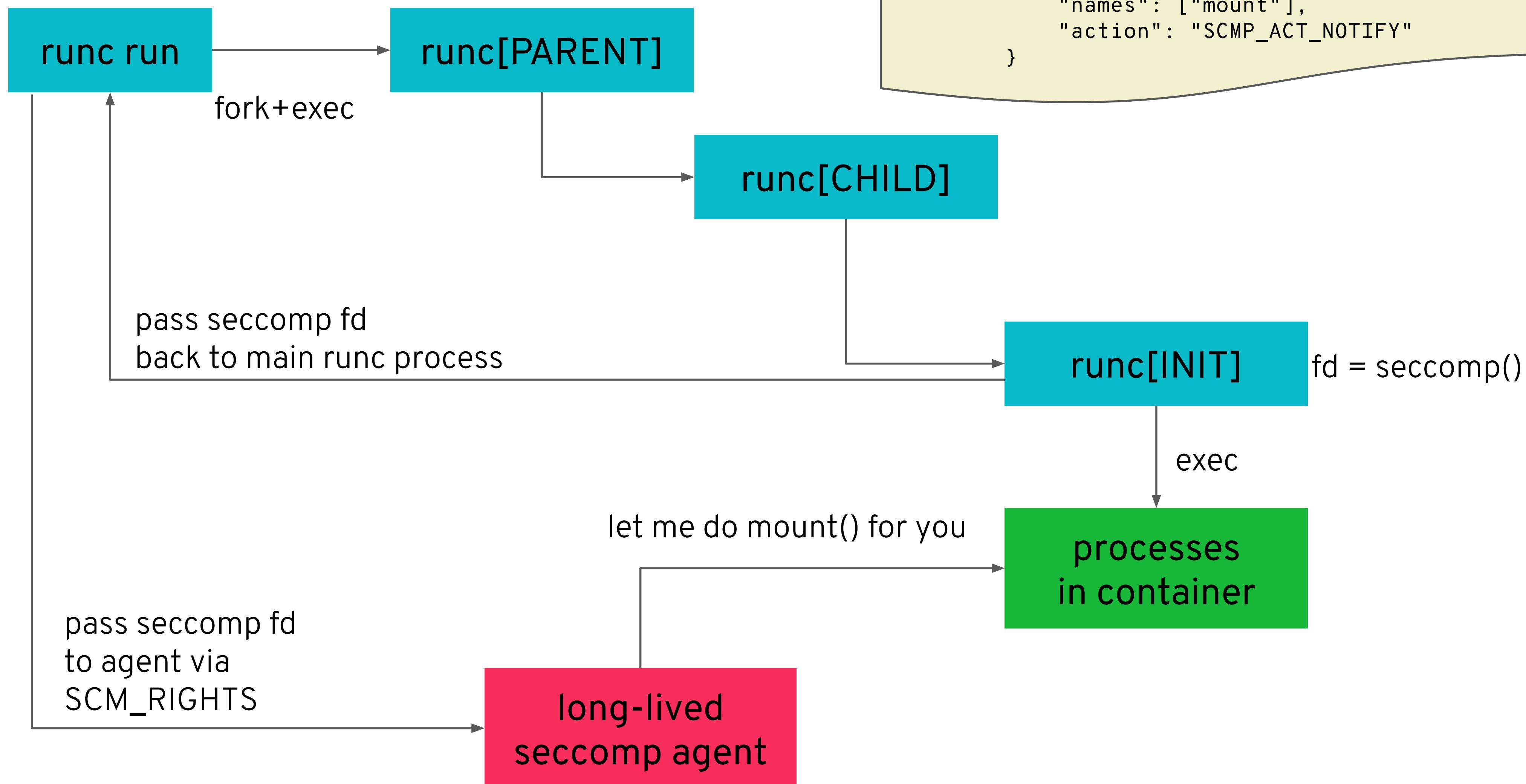


# Using a Seccomp Agent for privileged ops

- Currently, the workspace executes some privileged operations via gRPC methods to ws-daemon
  - **prepareUserNS()**  
mount(\$containerRootFS, “shiftfs”, “mark”)
  - **mountProc()**  
mount(“proc”, “/tmp/somewhere”, “proc”)
- Soon, runc will support Seccomp Notify for deferring some syscalls (e.g. mount) to a seccomp agent



# Seccomp Notify



# New tech in this space

- Kubernetes KEPs
  - KEP-127: Support User Namespaces  
<https://github.com/kubernetes/enhancements/pull/2101>
  - KEP-2033: Rootless mode  
<https://github.com/kubernetes/kubernetes/pull/92863>
- Rootlesskit:
  - <https://github.com/rootless-containers/>
  - Rootlesskit, Usernetes, slirp4netns, bypass4netns
- Seccomp Notify in Kubernetes
  - OCI Runtime Spec: <https://github.com/opencontainers/runtime-spec/pull/1074>
  - runc: <https://github.com/opencontainers/runc/pull/2682>
  - crun: <https://github.com/containers/crun/pull/438>
  - common: <https://github.com/containers/common/pull/190>
  - Seccomp Agent <https://github.com/kinvolk/seccompagent>

