
cgroups



Go package for creating, managing, inspecting, and destroying cgroups. The resources format for settings on the cgroup uses the OCI runtime-spec found [here](#).

Examples (v1)

Create a new cgroup

This creates a new cgroup using a static path for all subsystems under `/test`.

- `/sys/fs/cgroup/cpu/test`
- `/sys/fs/cgroup/memory/test`
- etc....

It uses a single hierarchy and specifies cpu shares as a resource constraint and uses the v1 implementation of cgroups.

```
1 shares := uint64(100)
2 control, err := cgroup1.New(cgroup1.StaticPath("/test"), &specs.LinuxResources{
3     CPU: &specs.LinuxCPU{
4         Shares: &shares,
5     },
6 })
7 defer control.Delete()
```

Create with systemd slice support

```
1 control, err := cgroup1.New(cgroup1.Systemd, cgroup1.Slice("system.
2     slice", "runc-test"), &specs.LinuxResources{
3     CPU: &specs.CPU{
4         Shares: &shares,
5     },
6 })
```

Load an existing cgroup

```
1 control, err = cgroup1.Load(cgroup1.Default, cgroups.StaticPath("/test"))
```

Add a process to the cgroup

```
1 if err := control.Add(cgroup1.Process{Pid:1234}); err != nil {  
2 }
```

Update the cgroup

To update the resources applied in the cgroup

```
1 shares = uint64(200)  
2 if err := control.Update(&specs.LinuxResources{  
3     CPU: &specs.LinuxCPU{  
4         Shares: &shares,  
5     },  
6 }); err != nil {  
7 }
```

Freeze and Thaw the cgroup

```
1 if err := control Freeze(); err != nil {  
2 }  
3 if err := control.Thaw(); err != nil {  
4 }
```

List all processes in the cgroup or recursively

```
1 processes, err := control.Processes(cgroup1.Devices, recursive)
```

Get Stats on the cgroup

```
1 stats, err := control.Stat()
```

By adding `cgroups.IgnoreNotExist` all non-existent files will be ignored, e.g. swap memory stats without swap enabled

```
1 stats, err := control.Stat(cgroup1.IgnoreNotExist)
```

Move process across cgroups

This allows you to take processes from one cgroup and move them to another.

```
1 err := control.MoveTo(destination)
```

Create subgroup

```
1 subCgroup, err := control.New("child", resources)
```

Registering for memory events

This allows you to get notified by an eventfd for v1 memory cgroups events.

```
1 event := cgroup1.MemoryThresholdEvent(50 * 1024 * 1024, false)
2 efd, err := control.RegisterMemoryEvent(event)
```

```
1 event := cgroup1.MemoryPressureEvent(cgroup1.MediumPressure, cgroup1.
    DefaultMode)
2 efd, err := control.RegisterMemoryEvent(event)
```

```
1 efd, err := control.OOMEventFD()
2 // or by using RegisterMemoryEvent
3 event := cgroup1.OOMEvent()
4 efd, err := control.RegisterMemoryEvent(event)
```

Examples (v2/unified)

Check that the current system is running cgroups v2

```
1 var cgroupV2 bool
2 if cgroups.Mode() == cgroups.Unified {
3     cgroupV2 = true
4 }
```

Create a new cgroup

This creates a new systemd v2 cgroup slice. Systemd slices consider “-” a special character, so the resulting slice would be located here on disk:

- /sys/fs/cgroup/my.slice/my-cgroup.slice/my-cgroup-abc.slice

```
1 import (
2     "github.com/containerd/cgroups/v3/cgroup2"
```

```
3     specs "github.com/opencontainers/runtime-spec/specs-go"
4 )
5
6 res := cgroup2.Resources{}
7 // dummy PID of -1 is used for creating a "general slice" to be used as
8 // a parent cgroup.
9 // see https://github.com/containerd/cgroups/blob/1
10 // df78138f1e1e6ee593db155c6b369466f577651/v2/manager.go#L732-L735
11 m, err := cgroup2.NewSystemd("/", "my-cgroup-abc.slice", -1, &res)
12 if err != nil {
13     return err
14 }
```

Load an existing cgroup

```
1 m, err := cgroup2.LoadSystemd("/", "my-cgroup-abc.slice")
2 if err != nil {
3     return err
4 }
```

Delete a cgroup

```
1 m, err := cgroup2.LoadSystemd("/", "my-cgroup-abc.slice")
2 if err != nil {
3     return err
4 }
5 err = m.DeleteSystemd()
6 if err != nil {
7     return err
8 }
```

Kill all processes in a cgroup

```
1 m, err := cgroup2.LoadSystemd("/", "my-cgroup-abc.slice")
2 if err != nil {
3     return err
4 }
5 err = m.Kill()
6 if err != nil {
7     return err
8 }
```

Get and set cgroup type

```
1 m, err := cgroup2.LoadSystemd("/", "my-cgroup-abc.slice")
2 if err != nil {
3     return err
4 }
5
6 // https://www.kernel.org/doc/html/v5.0/admin-guide/cgroup-v2.html#
   threads
7 cgType, err := m.GetType()
8 if err != nil {
9     return err
10 }
11 fmt.Println(cgType)
12
13 err = m.SetType(cgroup2.Threaded)
14 if err != nil {
15     return err
16 }
```

Attention

All static path should not include `/sys/fs/cgroup/` prefix, it should start with your own cgroups name

Project details

Cgroups is a containerd sub-project, licensed under the Apache 2.0 license. As a containerd sub-project, you will find the:

- Project governance,
- Maintainers,
- and Contributing guidelines

information in our [containerd/project](#) repository.