Clase 6-control groups, namespaces, docker

miércoles, 23 de abril de 2025 18:05

CHRoot se usa para aislar apps del resto del sistema
Cambia el directorio raiz de un proceso, cambia solo ese proceso y los hijos
Aislas un proceso para que no vea todo sino una parte que yo quiero

Control groups:
Todos los procesos tienen la misma prioridad, forma de otorgar prioridades es limitada
Existen algunas herramientas, pero no lo hacen del todo bien
Entonces se crean control grups. Es un pseudo file system que permite que se organicen en grupos jerárquicos que tengan una cantidad limitada de recursos.

- cgroups provee lo siguiente:
  - **Resource Limiting:** grupos no pueden excederse en la utilización de un recurso (tiempo de CPU, cantidad de CPUs, cantidad de memoria, I/O, etc.)
  - **Prioritization:** un grupo puede obtener prioridad en el uso de los recursos (tiempo de CPU, I/O, etc.)
  - **Accounting:** permite medir el uso de determinados recursos por parte de un grupo (estadísticas, monitoreo, billing, etc.)
  - **Control:** permite freezar y reiniciar un grupo de procesos

Los procesos no saben que están siendo usados en un control group

Dos versiones:
- V1: controlaores que se fueron agregando para controlar distintos tipos de recursos. Se complicó mucho, se fue diseñando después de implementr y era medio una kk.
- V2: trata de solucionar eso

Puedo tener las dos versiones al mismo tiempo, pero un controlador no puede estar en ambas a la vez.

CGROUP: asocia un conjunto de procesos con un conjunto de recursos.
Susistema: recursos que quiero limitar.
Jerarquia: la forma de acomodarse de los recursos

Cada jerarquia se define creando, eliminando y renombrando directorios dentro del pseudo filesystem
Cada proceso solo puede estar en un cgroup dentro de una jerarquia.

Obvio si el padre tiene ciertos recursos, los hijos tienen que ser iguales o de menos
Si un proceso crea un hijo, este pertenece al cgroup del padre.
Un controlador se puede desmontar si está libre, no tiene cgroups hijos
Cada cgroup fylesistem tiene un único raiz al que pertenecen todos los procesos

Controlador=recursos que voy a limitar
Control group=los grupos que limito, puedo tener varios dentro de cada controlador

Controlador= disco, memoria, etc.

En v1 habia una sola jerarquia
En v2 se monta una sola jerarquia. No permite montar uno solo en particular, si se monta se monta todo.

Agrega:
- Cgroup.controllers: archivo de lectura que dice qué recursos están habilitados para ese cgroup en particular.
- En group.subtree.control: se dice que recursos van a pasa a los hijos. Es de lectoescritura. El raiz

tiene todos,
Los procesos solo pueden estar en los procesos hijos (o en la raiz).
Se crea un archivo de lecto escritura cgroup.procs con los procesos de ese cgroup

Hay un croups.events(en ambas v)
Dice cuantos procesos asignados tiene este o algún hijo.
No puedo tener procesos a la misma altura que cgoups, quedan siempre en las hojas

Namespaces:
Permite hacerle creer a un proceso que está viendo que solo existe lo que le doy, aisla. Lo que está en un espacio de nombres no se puede ver por otro
Solo pueden usar lo que ven.
Cualquier modificación que le hago al recurso queda limitado a él, no afecta al sistema en general.
Un proceso puede estar en un solo espacio de nombres de cada tipo. Cuando el proceso termina el namespace se borra

Tres nuevas systems-calls:

- **clone():** similar al fork. Crea un nuevo proceso y lo agrega al nuevo namespace especificado. Su funcionalidad puede ser controlada por flags pasados como argumentos
- **unshare():** agrega el actual proceso a un nuevo namespace. Es similar a clone, pero opera en al proceso llamante. Crea el nuevo namespace y hace miembro de él al proceso llamador.
- **setns():** agrega el proceso actual a un namespace existente. Desasocia al proceso llamante de una instancia de un tipo de namespace y lo reasocia con otra instancia del mismo tipo de namespace

Permite que un proceso tenga dos pid distintos.
Cuando hablo de espacios de nombre, son procesos que se ejecutan en el sistema operativo común.Si un proceso hace un kernel panic se asustan todos
Todo se puede aislar.

Un *user namespace* permite que **un proceso tenga una identidad de usuario diferente dentro del namespace que fuera de él**. O sea:

- Dentro del contenedor (namespace), puede parecer que sos **root (UID 0)**.
- Pero desde el punto de vista del sistema host, sos **otro usuario sin privilegios** (por ejemplo, UID 1000).

Porque te permite **ejecutar procesos como root dentro de un contenedor, sin tener privilegios de root en el host**. Esto mejora la seguridad, ya que no se pueden hacer cosas peligrosas en el sistema real aunque parezca que tenés control total dentro del contenedor.

Contenedores:
Tecnologia liviana de virtualización a nivel so que permite ejecutar muchos sistemas aislados en un único host.
Las instancias ejecutan en el espacio de usuario, comparten =kernel.
Son procesos dentro del so base
Todos asociados al mismo kernel
El contenedor tiene todas las dependencias/librerias que necesita para ejecutarse.
Dos tipos de contenedores: de sistemas operativos, hago como una replica del so (duplico todo menos el kernel)
De aplicaciones: empaqueto una app en partivular, cada contenedro da un servicio. Cada contnedor hace algo, luego se comunican entre si.
Tener esto permite que microservicios evolucionen independientemente. Les da aislación

- Sus principales características:
  - Autocontenidos: tiene todo lo que necesita para funcionar
  - Aislados: mínima influencia en el nodo y otros contenedores
  - Independientes: administración de un contenedor no afecta al resto
  - Portables: desacoplados del entorno en el que ejecutan. Pueden ejecutar de igual manera en diferentes entornos.

Con control groups limito los espacios
El contenedor permite construir y empaquetar aplicaciones . Incluye codigo y librerias y dependencias para ejecutarse.
Ejecutan en un modo usuario con un kernel compartido.