pre-GDB
CVMFS remote snapshotter

Simone Mosciatti - EP-SFT CernVM(FS) team

Focus

Exploit CVMFS to host container images

- Already possible with singularity
- Works (all right) with docker + cvmfs graphdriver

Updates on containerd will make it simple and reliable to deploy more robust solution to all containerd based runtime (k8s).

Big pluses!

- Actually using the same image, not a different one like with the graphdriver plugin.
- Very simple code exploiting ktock work.
- We don’t pull the image anymore, we just mount it from CVMFS

High Level Overview

containerd plugin, code that need to run alongside the main runtime.

Workflow:

1. The remote-snapshotter looks for a layer.
2. It uses itself a plugin, the file system plugin, to deliver the layer
3. If the file system plugin finds the layer, it mounts it.
4. Otherwise, containerd falls back to downloading the layer from the registry.

Simple to blend with work already done in unpacked.cern.ch

How it works

```
+ func (fs *filesystem) Mount(ref, digest, moutpoint string) error {
+       digest = strings.Split(digest, ":")[1]
+       firstTwo := digest[0:2]
+       path := filepath.Join("/", "cvmfs", "unpacked.cern.ch", ".layers", firstTwo, digest, "layerfs")
+       if _, err := os.Stat(path); os.IsNotExist(err) {
+               fmt.Printf("layer not in the cvmfs repository: %s", digest)
+               return fmt.Errorf("layer %s not in the cvmfs repository", digest)
+       }
+       fmt.Printf("Found layer: %s\n", digest)
+       err := syscall.Mount(path, moutpoint, "", syscall.MS_BIND, "")
+       if err != nil {
+               fmt.Printf("%+v\n", err)
+       }
+       return err
+ }
```

How it works

```
+ func (fs *filesystem) Mount(ref, digest, moutpoint string) error {
+       digest = strings.Split(digest, ":")[1]
+       firstTwo := digest[0:2]
+       path := filepath.Join("/", "cvmfs", "unpacked.cern.ch", ".layers", firstTwo, digest, "layerfs")
+       if _, err := os.Stat(path); os.IsNotExist(err) {
+               fmt.Printf("layer not in the cvmfs repository: %s", digest)
+               return fmt.Errorf("layer %s not in the cvmfs repository", digest)
+       }
+       fmt.Printf("Found layer: %s\n", digest)
+       err := syscall.Mount(path, moutpoint, "", syscall.MS_BIND, "")
+       if err != nil {
+               fmt.Printf("%+v\n", err)
+       }
+       return err
+ }
```

Find the location of the layer in unpacked.cern.ch

How it works

```
+ func (fs *filesystem) Mount(ref, digest, moutpoint string) error {
+       digest = strings.Split(digest, ":")[1]
+       firstTwo := digest[0:2]
+       path := filepath.Join("/", "cvmfs", "unpacked.cern.ch", ".layers", firstTwo, digest, "layerfs")
+       if _, err := os.Stat(path); os.IsNotExist(err) {
+               fmt.Printf("layer not in the cvmfs repository: %s", digest)
+               return fmt.Errorf("layer %s not in the cvmfs repository", digest)
+       }
+       fmt.Printf("Found layer: %s\n", digest)
+       err := syscall.Mount(path, moutpoint, "", syscall.MS_BIND, "")
+       if err != nil {
+               fmt.Printf("%+v\n", err)
+       }
+       return err
+ }
```

Find the location of the layer in unpacked.cern.ch

If the layer is not there, return an error.

How it works

```
+ func (fs *filesystem) Mount(ref, digest, moutpoint string) error {
+       digest = strings.Split(digest, ":")[1]
+       firstTwo := digest[0:2]
+       path := filepath.Join("/", "cvmfs", "unpacked.cern.ch", ".layers", firstTwo, digest, "layerfs")
+       if _, err := os.Stat(path); os.IsNotExist(err) {
+               fmt.Printf("layer not in the cvmfs repository: %s", digest)
+               return fmt.Errorf("layer %s not in the cvmfs repository", digest)
+       }
+       fmt.Printf("Found layer: %s\n", digest)
+       err := syscall.Mount(path, moutpoint, "", syscall.MS_BIND, "")
+       if err != nil {
+               fmt.Printf("%+v\n", err)
+       }
+       return err
+ }
```

Find the location of the layer in unpacked.cern.ch

If the layer is not there, return an error.

Otherwise bind mount the layer

How it works

```
+ func (fs *filesystem) Mount(ref, digest, moutpoint string) error {
+       digest = strings.Split(digest, ":")[1]
+       firstTwo := digest[0:2]
+       path := filepath.Join("/", "cvmfs", "unpacked.cern.ch", ".layers", firstTwo, digest, "layerfs")
+       if _, err := os.Stat(path); os.IsNotExist(err) {
+               fmt.Printf("layer not in the cvmfs repository: %s", digest)
+               return fmt.Errorf("layer %s not in the cvmfs repository", digest)
+       }
+       fmt.Printf("Found layer: %s\n", digest)
+       err := syscall.Mount(path, moutpoint, "", syscall.MS_BIND, "")
+       if err != nil {
+               fmt.Printf("%+v\n", err)
+       }
+       return err
+ }
```

That is basically all.

Along with few boilerplate code.

Questions that remains open

- How to deploy, i.e. how does the plugin installation work with containerd standalone and with containerd in k8s?
- Regarding deployment, would it help if the plugin was gRPC based instead of a shared object so that we can create a plugin container with a pre-bundled cvmfs?
- Who would use it on the GRID or in the CERN data center? What would be the impact on unpacked.cern.ch? (# of images that we manage and ingest daily)