Open Container Initiative and
runc

Tomáš Tomeček, Software Engineer @ Red Hat
@TomasTomec

- main container-related experience comes from docker and kubernetes
- docker user since ~docker-1.0
- development lead of docker image build service @ Red Hat
- docker contributor (mainly docker-py and docker-compose)
- python developer, engineer, hacker, speaker
- created TUI for docker engine recently:
  - [https://github.com/TomasTomecek/sen](https://github.com/TomasTomecek/sen)
Open Container Initiative

- specification
- implementation
container formats

- **docker**
  - image format v1
  - image format v2
  - container format

- **rkt**
  - appc
    - App Container Image
    - App Container Image Discovery
    - App Container Pod
    - App Container Executor

- ...
libcontainer

- go library for spawning containers
- namespaces - `man 7 namespaces`
- cgroups
- capabilities - `man 7 capabilities`
- CRIU - “this is a project by various mad Russians to perform checkpoint/restore mainly from userspace”, Linus Torvalds
**runc**

- implementation of OCI spec
- features
  - starting container
  - checkpoint, restore (CRIU)
  - sending signals to container
  - runtime information about container
  - exec in container
  - generate spec
  - pause/resume

- `$ runc`
what does container engine add?

- registry and registry protocol
- a way to build images
- image format metadata
- seamless user interface
- networking
- volumes
- backend (graph)
- signing
<table>
<thead>
<tr>
<th>OpenShift</th>
<th>userspace</th>
</tr>
</thead>
<tbody>
<tr>
<td>kubernetes, swarm, compose</td>
<td></td>
</tr>
<tr>
<td>docker, rkt</td>
<td></td>
</tr>
<tr>
<td>runc</td>
<td></td>
</tr>
<tr>
<td>libcontainer</td>
<td></td>
</tr>
<tr>
<td>namespaces, cgroups, capabilities</td>
<td>kernel</td>
</tr>
<tr>
<td>kernel</td>
<td></td>
</tr>
<tr>
<td>hardware</td>
<td></td>
</tr>
</tbody>
</table>
spec - Filesystem Bundle

- set of files required to run a container in a runtime
- config.json - host independent (security, env vars, application args)
- runtime.json - host specific (mounts, hooks)
- directory with root filesystem - spec suggests rootfs
config.json sample - “version” and “root”

"version": "0.1.0",
"root": {
    "path": "rootfs",
    "readonly": true
},
config.json sample - “process”

"process": {
  "terminal": true,
  "user": {
    "uid": 0, "gid": 0, "additionalGids": null
  },
  "args": ["sh"],
  "env": [
    "PATH=/usr/bin:/sbin:/bin",
    "TERM=xterm"
  ],
  "cwd": ""
},
config.json sample - “mounts”

"mounts": [
  {
    "name": "proc", "path": "/proc"
  },
  {
    "name": "dev", "path": "/dev"
  },
  {
    "name": "sysfs", "path": "/sys"
  },
  {
    "name": "cgroup", "path": "/sys/fs/cgroup"
  }
],
config.json sample - “linux”-deependant configuration

"linux": {
    "capabilities": [
        "CAP_AUDIT_WRITE",
        "CAP_KILL",
        "CAP_NET_BIND_SERVICE"
    ],
},
"platform": {
    "os": "linux",
    "arch": "amd64"
}
config.json sample - “linux”-deependant configuration (2)

"uidMappings": [
  {
    "hostID": 1000,
    "containerID": 0,
    "size": 10
  }
],
"gidMappings": [
  {
    "hostID": 1000,
    "containerID": 0,
    "size": 10
  }
]
"hooks" : {
    "prestart": [
        {
            "path": "/usr/bin/fix-mounts",
            "args": ["arg1", "arg2"],
            "env": [ "key1=value1"
        },
        {
            "path": "/usr/bin/setup-network"
        }
    ]
}
"poststart": [  
  
  
  {  
    "path": "/usr/bin/notify-start"
  }
],

"poststop": [  
  
  
  {  
    "path": "/usr/sbin/cleanup.sh",
    "args": ["-f"]
  }
]
runtime.json - mounts

"mounts": {
    "proc": {
        "type": "proc",
        "source": "proc",
        "options": []
    },
    "dev": {
        "type": "tmpfs",
        "source": "tmpfs",
        "options": ["nosuid","strictatime",
                     "mode=755","size=65536k"]
    },
    "data": {
        "type": "bind",
        "source": "/volumes/testing",
        "options": ["rbind","rw"]
    }
}
runtime.json - linux-specific configuration

- namespaces
- devices (mknod)
- cgroups - extremely precise configuration of cgroups
- sysctl
- limits
- SELinux, apparmor
- seccomp
Conclusion

- runc is a framework for creating container engines
- runc is plumbing
- docker is porcelain
- runc is a precise low-level container definition and runtime
demo

- HAproxy + x instances of httpd