Goals for Today

- General orientation about how we (developers/sysadmin) use containers for development and deployment
- Differences between emulators, virtual machines, and containers
- Guided tour of DC-kube, our Kubernetes cluster

The Dependency Issue

- Software written for one computer architecture can’t be run on another
- All software require a specific set of libraries to run
  - Common that different software requires the same library but different versions

ADD OPERATION OPCODE

PERFORMING ADDITION USING DIFFERENT ARCHITECTURES

ARM	0100
MIPS	100011
x86	000000xy

EMULATORS

- Translate machine code
- Run software for x86 on ARM
- ”Slow”
- QEMU

Virtual Machines

- Abstract hardware
- Can be transferred between physical computers
- Requires dedicated hardware
- ”Long-term”
- QEMU
- KVM
- VirtualBox (x86 only)
- May run at almost the same speed as the host

Demo: QEMU and virtual machines

Containers

- Namespace isolation
- Sharing hardware with all other containers and the host
- Images — ”templates”
- ”Short-term”
- No host — runs normally on the computer

Containers

Software

- Open Container Initiative (OCI)
- Repositories
  - Dockerhub
  - Github Container Repository

- Docker
- Podman
- Containerd
- Singularity
- LXC

Demo

Docker Desktop/Rancher Desktop

Using a container

Dockerfile

Docker-compose

KUBERNETES

- Container orchestration
- Originates from Google
- ”docker-compose for clusters”

Demo: DC-Kube