Introduction to Kubernetes
Containers
container vs virtual machine

Virtual machine
runs its own kernel

Container
uses the host kernel
containers are not secure

It is possible to execute code on a host system from a container
Containers are weird

Container operating system is a hybrid OS: it pairs arbitrary host kernel with arbitrary containerized user-space.
takeaways

- Do not let unprivileged users in
- Do not rely on kernel or even system features

But since you will anyway,

test containers with the production-like host kernel
FROM debian:jessie
RUN apt-get update \
    && apt-get install gradle \
    && apt-get clean
RUN apt-get update \
    && apt-get install \
    protobuf-compiler \
    && apt-get clean

Dockerfile reference

..with protoc
..with gradle
debian:jessie
advanced topics

• Volumes
  You can mount directories from a host system into container
  Ownership and permissions on files are preserved
  => Problem with deleting files owned by root

• Networking
  You can connect port on the host system with the port exposed by container

  There is more, but “It’s complicated”
Continuous Integration
with containers
containers for ci

At build

- Take base image
- Add build tools
- Add dependency cache
- Add git cache

Runtime

- Update git cache
- Run build task
- Publish artifact to storage
- Discard container

Git repo is a source, application is an artifact, container is a build tool
containers for prod

At build

- Take base image
- Inject app artifact
- Define endpoint

Runtime

- Run container with exposed endpoint

Application is a source, container image is an artifact
lifecycle

GIT
Bitbucket

code

Maven repo
Artifactory*

JAR

gradle build

docker build

docker image

Registry
Nexus

code

JAR

JAR
Kubernetes
There is a registry with Docker images
now what?
Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.
vocabulary

- Image
- Container
  - runs image
- Pod
  - group of containers
- Node
  - host machine for pods
- Cluster
  - set of nodes

Applies to group of pods:

- Replica Set
  - replica counter
- Deployment
  - deployment strategy
- Service
  - common interface
deployment

replica set

service
kubernetes client tools

- Kubernetes provides full-featured REST API
- `kubectl` – command-line utility with full access to the API
  - Very verbose with a lot of subcommands and options
  - Resources can be created and managed directly by typing CLI commands
  - Or can be stored in YAML files and applied in batches by running
    $$\text{kubectl create -f <directory}>$$
- `kubernetes-dashboard` – an optional plugin which provides graphical web-interface with limited functionality
networking

- Every node has external IP address
- Every pod has internal IP address and domain name
- Every service has internal IP address and domain name

=> Pods can talk to pods and services by name or IP
But how do I reach the pod from the outside?
debugging pod

Pod with one container, based on “busybox” container image
When started, runs shell and waits for input
Works as a gate to internal network

(laptop)$ kubectl run -i --tty busybox --image=busybox --rm --restart=Never
Waiting for pod default/busybox-kz38w to be running, status is Pending, pod ready: false
Waiting for pod default/busybox-kz38w to be running, status is Pending, pod ready: false

Hit enter for command prompt
/ # telnet pod-1.local 8080
exposed service

NodePort – the same port on every node

- Assigned to the service
- Open for external network
- Traffic is redirected to one of the pods behind the service

192.168.42.152:9999
192.168.42.154:9999
10.123.15.2
pod-1.local
10.123.15.3
pod-2.local
10.123.15.3
pod-3.local
Demo