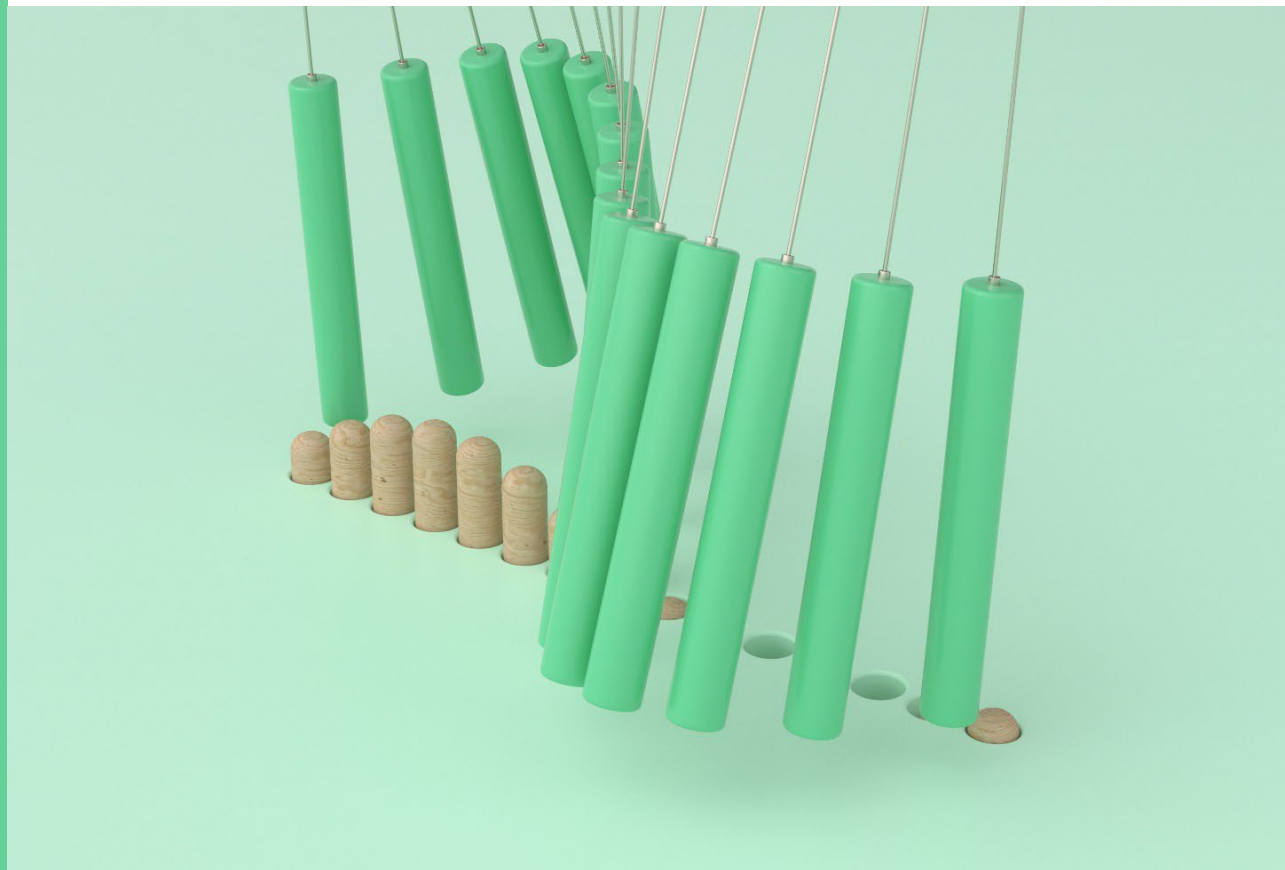
Docker Containers and Kubernetes Introduction

NetApp



David Fox

What Exactly are Containers?

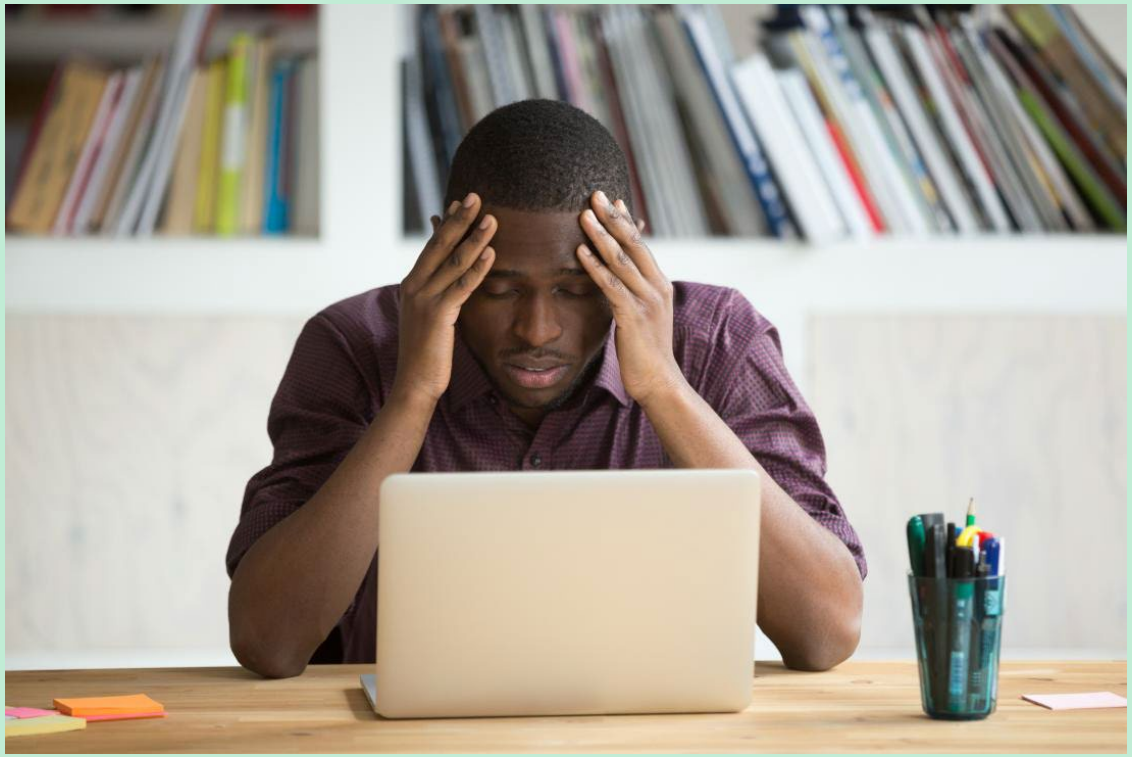
- A way to distribute applications as immutable images
- Leverages the Linux kernel of the container host
- Uses Linux cgroups to isolate containers from other containers & the system

How Do Containers Differ from Virtual Machines?

- More light weight
- Immutable
- Repeatable
- Easier to maintain
- Easier to deploy

Application Deployment before Containers

- Get a server
- Set up and maintain the server specific to the application
- Server is a mutable resource
  - Admins can login and make changes
- Manual deployment and configuration of application on top of server
  - Automation such as Ansible, Puppet or Chef can help
- This is hard to maintain!!



Application Deployment with Containers

- Get a server
- Set up server to run containers (i.e deploy a container runtime such as Docker)
- Run containers!
- Easy to maintain!!

Building a container

- The Dockerfile contains instructions on building the container
  - Base image
  - Package updates
  - Installing the application
  - Configuring the application
- Can be pushed to a Docker image repository for reuse after container image is built

```
FROM elixir:1.11.4-alpine

ENV UID=911 GID=911 \
    MIX_ENV=prod

ARG PLEROMA_VER=develop

RUN apk -U upgrade \
    && apk add --no-cache \
    build-base \
    cmake \
    git \
    file-dev

RUN addgroup -g ${GID} pleroma \
    && adduser -h /pleroma -s /bin/sh -D -G pleroma -u ${UID} pleroma

USER pleroma
WORKDIR /pleroma

RUN git clone -b develop https://git.pleroma.social/pleroma/pleroma.git /pleroma \
    && git checkout ${PLEROMA_VER}

COPY config/secret.exs /pleroma/config/prod.secret.exs

RUN mix local.rebar --force \
    && mix local.hex --force \
    && mix deps.get \
    && mix compile

VOLUME /pleroma/uploads/

CMD ["mix", "phx.server"]
```

Running the container

- Specify the image you want to use
- Pass in parameters
  - Network ports to bind to
  - Volumes to use
  - Environment variables
  - Container capabilities

```
→  ~ ls -al /home/cloud-user/tmp
total 36
drwxrwxr-x.   3 cloud-user cloud-user   121 Feb 16 16:23 .
drwxr-xr-x. 110 cloud-user cloud-user 16384 Feb 16 16:24 ..
-rw-r--r--.   1 cloud-user cloud-user   756 Jan  6 01:14 config_kerberos.zip
-rw-r--r--.   1 cloud-user cloud-user   106 Jan  6 19:11 install.yml
-rw-r--r--.   1 cloud-user cloud-user   382 Jan  6 01:14 install_yum_packages.zip
-rw-r--r--.   1 cloud-user cloud-user  2829 Jan  6 01:14 requirements.txt
drwxr-xr-x.   4 cloud-user cloud-user    57 Jan  6 19:10 roles
→  ~ sudo docker run -v /home/cloud-user/tmp:/mnt -it --rm busybox
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
/ # ls -al /mnt
total 16
drwxrwxr-x    3 301      301            121 Feb 16 16:23 .
dr-xr-xr-x    1 root     root            74 Feb 16 16:24 ..
-rw-r--r--    1 301      301            756 Jan  6 01:14 config_kerberos.zip
-rw-r--r--    1 301      301            106 Jan  6 19:11 install.yml
-rw-r--r--    1 301      301            382 Jan  6 01:14 install_yum_packages.zip
-rw-r--r--    1 301      301           2829 Jan  6 01:14 requirements.txt
drwxr-xr-x    4 301      301             57 Jan  6 19:10 roles
/ #
```

Containers – The New Challenge

- How do I deal with server failures?
- Is there a way to set up networking automatically?
- What about my data?
- What about if the container terminates for some reason – how can I monitor and automatically restart?
- How do I scale out?

Kubernetes - Intro

- A system to automatically schedule containers on worker nodes
- Provides a networking layer
- Presents data needed by containers no matter what host they live on
- Scale-out capabilities
- Will restart containers if they terminate for some reason

Kubernetes – Deploying Applications

- Kubernetes creates resources based on user submitted manifests in either JSON or (more commonly) YAML
- These manifests can create
  - Container deployment itself
  - Data volumes
  - Networking components such as load balancers and ingress
  - Scheduled tasks

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
  labels:
    app: go-webserver-ext
  name: go-webserver-ext
  namespace: default
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: go-webserver-ext
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: go-webserver-ext
    spec:
      containers:
        - image: 'docker.io/iafeoktistov/go-webserver:1.0'
          imagePullPolicy: IfNotPresent
          name: go-webserver
          ports:
            - containerPort: 8080
              protocol: TCP
          resources: {}
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```

Kubernetes – Maintaining Deployments

- Tune in to the next Partner Webinar for more about this!