Containers Make Things Easy, Right?

MAN, DOCKER IS BEING USED FOR EVERYTHING. I DON'T KNOW HOW I FEEL ABOUT IT.

ONCE, LONG AGO, I WANTED TO USE AN OLD TABLET AS A WALL DISPLAY.

I HAD AN APP AND A CALENDAR WEBPAGE THAT I WANTED TO SHOW SIDE BY SIDE, BUT THE OS DIDN'T HAVE SPLIT-SCREEN SUPPORT. SO I DECIDED TO BUILD MY OWN APP.

#redhat #rhsummit
Containers Make Things Easy, Right?

I downloaded the SDK and the IDE, registered as a developer, and started reading the language's docs.

...then I realized it would be way easier to get two smaller phones on eBay and glue them together.

But you never learned to write software.

No, I just learned how to glue together stuff that I don't understand.

I...OK, Fair.

On that day, I achieved software enlightenment.
AGENDA

- Capabilities, Problems, and Trade offs
- Fundamental Shift in Mindset
- Implications & Common Obstacles
  - (And how to overcome them!)
- The Tenets of Building
- Putting It All Together
CAPABILITIES, CHALLENGES, AND TRADE OFFS
“Most uses of Docker are like a junk drawer: neat on the outside, a total mess on the inside. People stuff their python 2 app in there and forget what their dependencies are, or where they got them from. Good luck upgrading that 2-3 years from now.”

- gerbilly on Hacker News
“Using containers is as much of a business advantage as a technical one. When building and using containers, layering is crucial. You need to look at your application and think about each of the pieces and how they work together—similar to the way you can break up a program into a series of classes and functions.”

- Ryan Hallisey
BLOCKERS

1. Code: mysqld
2. Configuration: /etc/my.cnf
3. Data: /var/lib/mysql
4. Other stuff :-)

#redhat #rhsummit
### OTHER STUFF :-)!

<table>
<thead>
<tr>
<th></th>
<th>EASY</th>
<th>MODERATE</th>
<th>DIFFICULT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Code</strong></td>
<td>Completely Isolated (single process)</td>
<td>Somewhat Isolated (multiple processes)</td>
<td>Self Modifying (e.g. Actor Model)</td>
</tr>
<tr>
<td><strong>Configuration</strong></td>
<td>One File</td>
<td>Several Files</td>
<td>Anywhere in Filesystem</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>Saved in Single Place</td>
<td>Saved in Several Places</td>
<td>Anywhere in Filesystem</td>
</tr>
<tr>
<td><strong>Secrets</strong></td>
<td>Static Files</td>
<td>Network</td>
<td>Dynamic Generation of Certificates</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>HTTP, HTTPS</td>
<td>TCP, UDP</td>
<td>IPSEC, Highly Isolated</td>
</tr>
<tr>
<td><strong>Installation</strong></td>
<td>Packages, Source</td>
<td>Installer and Understood Configuration</td>
<td>Installers (install.sh)</td>
</tr>
<tr>
<td><strong>Licensing</strong></td>
<td>Open Source</td>
<td>Proprietary</td>
<td>Restrictive &amp; Proprietary</td>
</tr>
<tr>
<td><strong>Recoverability</strong></td>
<td>Easy to Restart</td>
<td>Fails Sometimes</td>
<td>Fails Often</td>
</tr>
</tbody>
</table>
APPLICATION DELIVERY

Container images, assembly instructions, and resource requirements
PRODUCTION-READY CONTAINERS

More than just container images to deliver real applications

- Image Build Instructions
  - Source Control
- Container Images
  - Registries
- Orchestration Definitions
  - Source Control
- Operators
  - Operators Lifecycle Manager

```yaml
apiVersion: v1
type: ReplicationController
metadata:
  name: mysql
labels:
  name: mysql
spec:
  replicas: 1
template:
  metadata:
    name: mysql
  spec:
    containers:
      - name: mysql
        image: openshift3/mysql-55-rhel7
        env:
          - name: MYSQL_ROOT_PASSWORD
            value: pizza
          - name: MYSQL_USER
            value: pizza
          - name: MYSQL_PASSWORD
            value: pizza
```
SOFTWARE NIRVANA

How will you use your extra free time?
The Tenets of Building
THE 5 COMMANDMENTS

Foundational to all of these rules is source control for everything - treat all of the artifacts as buildable from code

- Standardize
- Minimize
- Delegate
- Process
- Iterate
STANDARDIZE

**Goal:** Publish a standard set of images with common lineage

- **Base image(s)**
  - Application Frameworks
  - Application Servers
  - Databases
  - Etc

- **Benefits:**
  - Easier scale
  - Maximize reuse of common layers
  - Minimize pulls
  - Limit environment anomalies
**Goal:** Limit the content in the image to what serves the workload

- buildah can populate images with tools from the host.

**Benefit:**
- Smaller attack / patching surface
- More efficient push/pulls

**Warning:** taking this to the extreme will negate layer sharing and not have the intended effect.

---

# Minimize

- Build OCI/docker Images
- Leverage the host tools
- Leverage a buildah container

---

@redhat @rhsummit
Goal: Ownership needs to lie with expertise

Benefit: Leverage your teams on the part of the stack they know best
FOCUS ON PROCESS AND AUTOMATION

**Goal**: Automate rebuilds of all objects
- Testing (CI, performance, etc)
- Security
- Operators

**Benefits**: Fast redeployment as you make changes to the environment
**Goal:** DON’T REPEAT THE MISTAKES OF THE PAST!!!!!

**Benefit:** Capture it in code. Knowledge is temporal.
3 IN A ROW!

THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"My Docker containers are building!"

HEY! GET BACK TO WORK!

Docker!

OH, CARRY ON.
PUTTING IT ALL TOGETHER
DON’T GET STUCK IN A SINGLE NODE MINDSET

This mindset:
- Only thinks about container images
- Treats containerized applications like traditional applications
- Doesn’t fully leverage the power of containers
- Doesn’t think about automation at day two

Traditional Development

FIND  RUN  BUILD  SHARE
THINK ABOUT DAY-2 OPERATIONS

This mindset:
● Think about how everything can be automated
● Offload updates, backups, restarts, failures all mundane tasks
● Interact through an API, not by SSH’ing into nodes
● Drive the entire platform by defining state, not just the applications
ANOTHER HILARIOUS XKCD

"I SPEND A LOT OF TIME ON THIS TASK. I SHOULD WRITE A PROGRAM AUTOMATING IT!"

**THEORY:**

- Writing Code
- Work on Original Task
- Automation Takes Over
- Free Time

**REALITY:**

- Writing Code
- Debugging
- Ongoing Development
- Rethinking
- No time for original task anymore

#redhat #rhsummit
THANK YOU

linkedin.com/company/Red-Hat
youtube.com/user/RedHatVideos
facebook.com/RedHatInc
twitter.com/RedHat
WHAT CHALLENGES DO CONTAINERS REALLY SOLVE IN PRODUCTION?

True
- Better separation of concerns between developers, operations, database administrators, middleware specialists, etc
- Compatibility and portability still need to be planned for.
- Developers and operations need a mix of new and existing skills
- Better definitions of applications & sub-components
- Truly distributed systems environment

False
- Everybody can do whatever they want. Developers will just do everything themselves. We no longer need specialists.
- Complete portability - build once, run anywhere. I...mean...anywhere
- Containers are easy. Developers just use them, don’t worry...
- You must completely break your application up
- Forget everything you know, this is magic