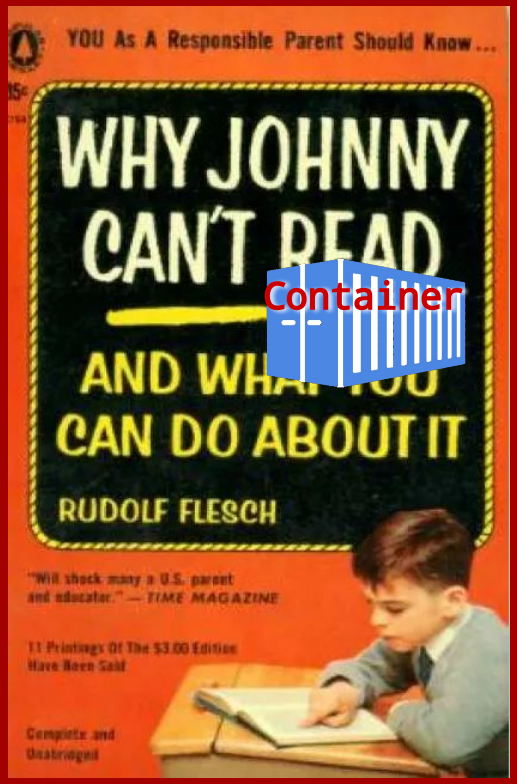


Why Johnny can’t Container

Louis P. Santillan
Architect, Container and PaaS Practice
Mar 2018

What's with the title?

- "Why Johnny Can't Read" was a critique of American "sight word" style reading and an advocation for "phonics" style reading
- In my opinion, Flesch advocated teaching children "how to learn" to read over memorization
- I feel the problem is similar with people learning to develop on Container Platforms

What do you reach for first?

Google?

Stack Overflow?

Github?

Coding by Braille

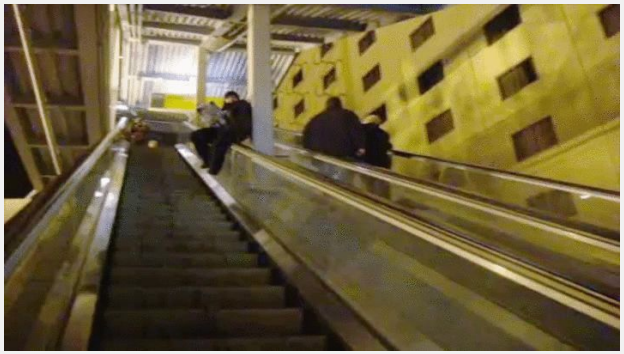
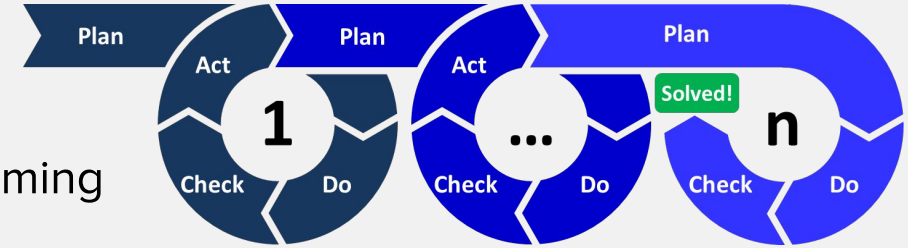


Image credits to Giphy and/or their respective creators

Know Your References

W. Edwards Deming



The Deming Cycle: Plan - Do - Check - Act

- “In God we trust; all others bring data.”
- “Without data, you're just another person with an opinion.”
- “A bad system will beat a good person every time.”

https://en.wikipedia.org/wiki/PDCA

Know Your References

The Unix Philosophy - The McIlroy Definition

- “Write programs that do one thing and do it well.”
- “Write programs to work together.”
- “Write programs to handle text streams, because that is a universal interface.”

https://en.wikipedia.org/wiki/Unix_philosophy#Origin

Know Your References

The Suckless Philosophy

- Minimalism produces higher quality, more performant, more secure software
- “Code complexity is the mother of bloated, hard to use, and totally inconsistent software. With complex code, problems are solved in suboptimal ways, valuable resources are endlessly tied up, performance slows to a halt, and vulnerabilities become a commonplace.”

https://suckless.org/philosophy

Know Your References

DJB - Daniel J. Bernstein

- Minimize trusted code and design clean interfaces
- “Minimizing privilege is not the same as minimizing the amount of trusted code, and does not move us any closer to a secure computer system... The defining feature of untrusted code is that it cannot violate the user’s security requirements.”

http://cr.yp.to/djb.html

https://cr.yp.to/qmail/qmailsec-20071101.pdf

Know Your References

THE TWELVE-FACTOR APP

INTRODUCTION

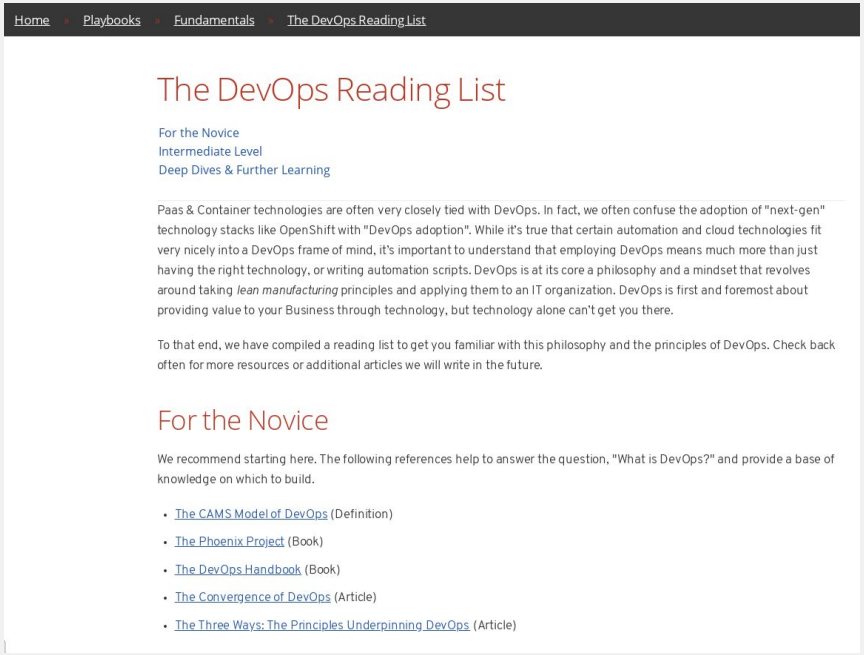
In the modern era, software is commonly delivered as a service: called *web apps*, or *software-as-a-service*. The twelve-factor app is a methodology for building software-as-a-service apps that:

- Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;
- Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration;
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility;
- And can **scale up** without significant changes to tooling, architecture, or development practices.

The twelve-factor methodology can be applied to apps written in any programming language, and which use any combination of backing services (database, queue, memory cache, etc).

https://12factor.net

Know Your References



Home » Playbooks » Fundamentals » The DevOps Reading List

The DevOps Reading List

For the Novice
Intermediate Level
Deep Dives & Further Learning

Paas & Container technologies are often very closely tied with DevOps. In fact, we often confuse the adoption of "next-gen" technology stacks like OpenShift with "DevOps adoption". While it's true that certain automation and cloud technologies fit very nicely into a DevOps frame of mind, it's important to understand that employing DevOps means much more than just having the right technology, or writing automation scripts. DevOps is at its core a philosophy and a mindset that revolves around taking *lean manufacturing* principles and applying them to an IT organization. DevOps is first and foremost about providing value to your Business through technology, but technology alone can't get you there.

To that end, we have compiled a reading list to get you familiar with this philosophy and the principles of DevOps. Check back often for more resources or additional articles we will write in the future.

For the Novice

We recommend starting here. The following references help to answer the question, "What is DevOps?" and provide a base of knowledge on which to build.

- The CAMS Model of DevOps (Definition)
- The Phoenix Project (Book)
- The DevOps Handbook (Book)
- The Convergence of DevOps (Article)
- The Three Ways: The Principles Underpinning DevOps (Article)

http://v1.uncontained.io/playbooks/fundamentals/devops_reading_list.html


2 Pronged Attack
Tactical & Strategic
redhat.

The Tactical

- Tips, Tricks, and Sources to Remember
- Start with a checklist!
- If you don’t have one, you can borrow mine!

My Checklist

1. Pick a base image which is closest to your application platform or framework
2. Use tools that make building your container easier
   Use s2i in OpenShift and OpenShift Origin
   Use a CI/CD tool like Jenkins
   Use Templates for describing your app
3. Understand how to use your base image to build your app
4. Pull out (Parameterize) all of the config from internals of your app
5. Configure your logging to write everything to stdout

Parameterizing your Configuration

- WHY? - Build Once, Deploy Anywhere
- HOW? - Have App startup/initialization based on external factors and information
- Your code shouldn’t need to be rebuilt for DEV or QA or PROD
- Be comfortable with creating ENV vars, ConfigMaps, Secrets, Volume Mounts

Using s2i

If you're using s2i, standing your app up (once you've built your binary) can be as simple as

```
    # Create a new "bare" build (creates BuildConfig (bc) & ImageStream (is))
    $ oc new-build --image-stream=openshift/jboss-xxxx --binary=true --name=app
    # Define a Deployable App (creates DeploymentConfig (dc) and Service (svc))
    $ oc new-app app
    # Let the app become accessible to the outside world
    $ oc expose svc/app
    # Restart the build using the WAR file dropped in
```

Example: JBoss EAP with Binary s2i Deploy

Rebuilding/redeploying the app is as simple as

```
$ mvn clean package && \
    cp -rv target/* s2i/deployments/
$ oc start-build app --from-dir=s2i/
```

Have the following folder structure becomes **/opt/eap** in the

$app/s2i/deployments	WAR, JARS, EARS
$app/s2i/configuration	standalone-openshift.xml, properties files
$app/s2i/modules	*MQ Drivers, JDBC Drivers, etc.

In the container, the **$app/s2i** folder becomes **/opt/eap**

https://access.redhat.com/documentation/en-us/red_hat_jboss_middleware_for_openshift/3/html-single/red_hat_java_s2i_for_openshift/#binary_builds

More JBoss EAP, Tomcat/JWS Tips

- Your apps must run in Standalone Mode; Domain Mode is not usable
- Clustering is done via JGroups instead
- Be sure your pom.xml does not create doc or source jars
- JNDI connections should be externalized in standalone-openshift.xml
- Property vars with `.` need to be renamed with underscores

  `APP.NAME` => **`APP_NAME`**

- If you need to overwrite startup command line parameters, use the right ENV var

  JBoss EAP - **`JAVA_OPTS_APPEND`**

  Tomcat/JWS - **`CATALINA_OPTS_APPEND`**

More JBoss EAP, Tomcat/JWS Tips

- To use ENV vars in your **`standalone-openshift.xml`** config file
  `${env.MYAPP_VAR}`
- Need a non-system JDK? Use `**`/usr/sbin/alternatives`**` to install and manage your JDK config

https://access.redhat.com/solutions/3190862

https://access.redhat.com/solutions/2107431

https://access.redhat.com/documentation/en-us/jboss_enterprise_application_platform/6.3/html/administration_and_configuration_guide/configure_the_default_jdk_on_red_hat_enterprise_linux

JBoss EAP, Tomcat/JWS References

https://access.redhat.com/documentation/en-us/red_hat_jboss_middleware_for_openshift/3/html-single/red_hat_jboss_enterprise_application_platform_for_openshift/#configuration_environment_variables

https://access.redhat.com/documentation/en-us/red_hat_jboss_middleware_for_openshift/3/html/red_hat_jboss_web_server_for_openshift/before_you_begin#comparison_jws_and_jws_for_openshift_image

https://access.redhat.com/documentation/en-us/openshift_enterprise/3.0/html-single/using_images/#differences-between-the-jboss-a-mq-xpaas-image-and-the-regular-release-of-jboss-a-mq

Strategic
Guidance for your Enterprise

The Strategic

- Measure!
- Use a Checklist to standardize On-boarding Activities
  - Take Inventory of the Current Development/Deployment Lifecycle
  - Take Inventory of your Candidate On-boarding Applications
  - Start with simple apps
- Create an Expert On-boarding Group

Measure

- What are you attempting to optimize?
  Development Cycle?
  Deployment Cycle?
  Quality?
  Standardization?

The Strategic Checklist

1. This checklist should allow you to take inventory of your Application's Architecture
   Ports, Dependencies, TLS, Startup, Monitoring, HA, etc.

2. Prioritize based on Value to the Business

3. Don't be afraid start with simple apps

4. Create Templates for your Technology Stacks

Create an Expert On-boarding Group

- Train the Trainers
- Create Internal Experts based on
  - General on-boarding skill
  - Technology Stack Focus
- Develop a way to disseminate knowledge

  Templates, Tutorials, Training, Wikis, etc.

Charlie and the Chocolate Factory

We are the music makers, and, we are the dreamers of dreams.

Copyright Warner Bros.

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/in/lpsantil

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

lsantill@redhat.com