The Mythical App Container

Different perspectives

Robert Schweikert
Public Cloud Architect
rjschwei@suse.com

What to expect ...

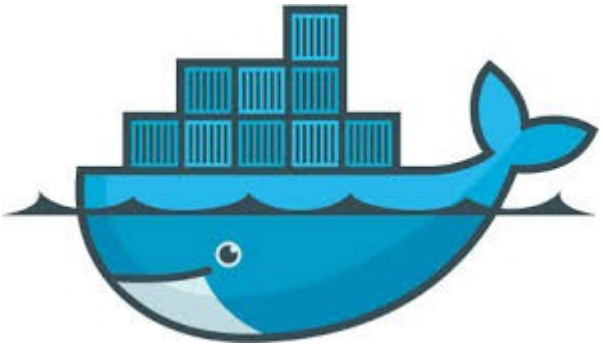
- I will
  - probably rattle the cage a little bit
  - most likely make some controversial statements
  - highlight some issues that do not get a lot of press
  - hopefully get/keep you thinking about this container stuff
    - Maybe from a bit of a different point of view
- Information about
  - somewhat unpleasant realities
  - the human condition

Some things to keep in mind….

- Old code never dies
  - There are still plenty of Cobolt developers
  - HP many years ago ported VMX to Itanium
- It takes tremendous effort to initiate change
  - Changing exiting practices in business is very expensive
  - But doing new things in new ways is not too difficult

Lets get started

- Docker is not the only game in town
  - Other container “engines” exist
- But Docker
  - gets the most hype (press)
  - has largest contributor base
  - made containers sexy
  - arguably made containers easy to use
  - is more than just a container “engine”
- Thus I will primarily focus on Docker



Developers love docker

- What's not to love?
  - Containers start fast
  - Easy to map working directory into the container
  - Container provides an unpolluted test environment every cycle
  - Runs on laptop, desktop, server the same way
  - No outside help (IT) needed
    - Unless of course you do not have root access to your machine
  - Easy to test a number of “distributions” if.....

Developers love docker

- But developers are a special species

The real world looks different

Poodle is of course different (protocol issue)

So then...

- What's inside is very important
  - Is it really just the application and the dependencies?

Well, not really....

- Init system
  - One can argue that the application itself does not depend on the init system
- Web apps need SSL
  - One can argue that the application itself does not depend on SSL
- What about
  - configuration management
  - service discovery
- Update stack
  - Containers generally contain a package management system

Configuration management?

- "Simply rebuild your container"
  - Implies that there is automation for configuration present
  - Container possibly needs to
    - connect to others
    - have access to external storage
    - have "non-island" characteristics

Some kind of automation is necessary

even if it is via active discovery, probably not built into the app

Update stack?

- Note
  - The current package systems are probably not the end all, but they comprise the predominant system in use. Code has to be delivered somehow.
- Needed to build a container
  - Could be removed at the end of the build process

Lets just say

The Application Container is a Myth

- Or at the very least a very very very loose interpretation of what dependencies mean
  - In the sense that the weather at the North Pole is dependent on a butterfly flapping it's wings in Timbuktu

Or…..

- Put only the application in the container
  - Use libraries as provided by the host

Application portability just went out the window

I think we can agree that

What's inside is very important

and therefore,

it is very important where you get it from


BILL OF MATERIAL

ITEM NO.	DESCRIPTION	UNIT	ASSEMBLY OR FSN NO.	QUANTITIES	
				TROP	NORTH

Thus we need...

- A standard that
  - declares the content in a way that is vendor neutral
    - The Container Spec initiative currently only declares how to handle the container
  - allows us to tie the content back to CVE system
  - allows us to manage the inside of the container
    - hmm this sounds a lot like regular systems management ….
    - but there is another trick, more on this in a bit ….
- Is this a pipe dream?
  - probably
    - OCI: "...for the express purpose of creating open industry standards around container formats and runtime…."

A look at the current realities

- Every distribution
  - applies their own patches
  - it takes time for those to migrate upstream
  - some may never go upstream
  - has different naming and version conventions
- How would we encode the differences?

What about ditching the distros?

- Do you really want to
  - download everything from upstream directly?
  - compile test and verify everything on your own?
  - stay on top of every patch that fixes a vulnerability?
- You'd probably not be part of the embargo list
  - Thus you'd always be behind with application of critical fixes

We can probably agree that

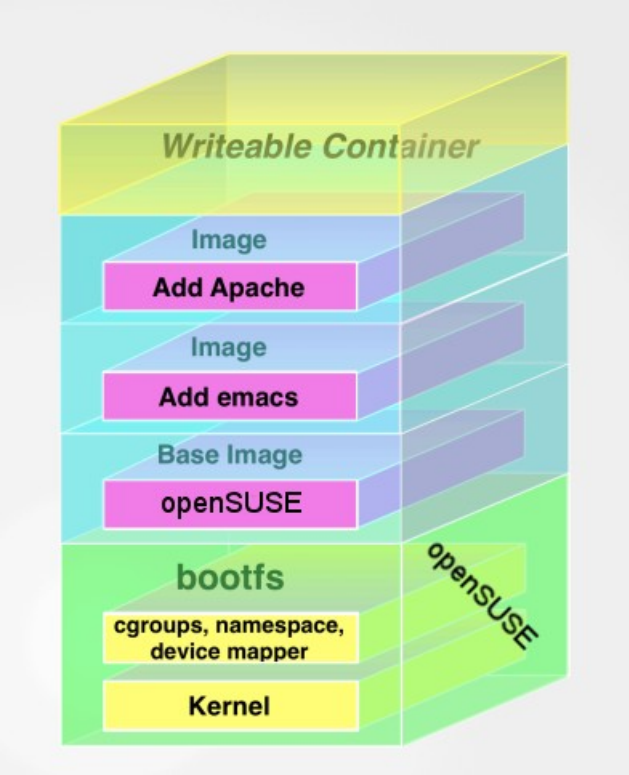
Distributions provide value, solve more problems than they create

Thus we have

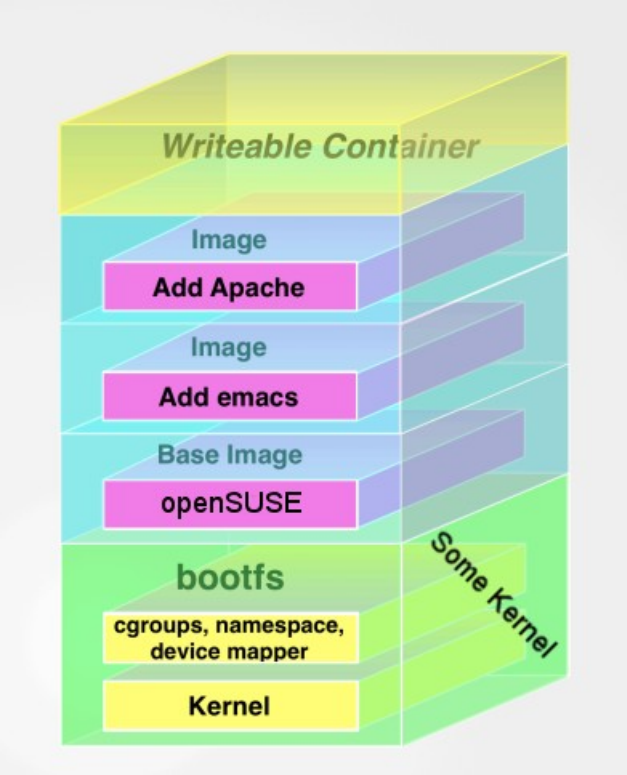
- A packaging system, containers, around/atop a packaging system, RPM/deb
- But the container does not
  - extend existing technology
  - in some ways works contrary to the intend of existing system
  - does not provide a BOM
- Things to work out

A quick detour

- Pop quiz:
  - What does it mean to run a given distribution in a container world?



Are we running openSUSE?



Back to what we need

- No standard, no problem, use a trusted vendor
  - Container from application vendor (ISV)
    - Gets us back to the world of today:
      - Vendor is responsible for everything, appliance model
      - I have to manage the container but there is more to this ….(be patient)
    - ISVs generally do not want to take a “full system” responsibility
    - End customers often don't feel comfortable handing “full system” responsibility over to a third party
      - This is changing with SaaS
      - But IaaS is really successful, primary driver of Public Cloud

The other option - trusted vendor

- No standard, no problem, use a trusted vendor
  - Container from distribution vendor
    - Generally do not want to have anything to do with application setup, especially for non open source ISV applications
  - Container from an integrator
    - Probably the best viable option

Or, roll your own

- Build your own containers
  - Control what goes in it
    - You still need code
  - Reasonably easy with Docker tools or KIWI
  - Delivers the BOM
- But we are back to the same problem
  - The application has to be installed
  - As builder I have to worry about library versions
    - At least part of the application portability promise is lost

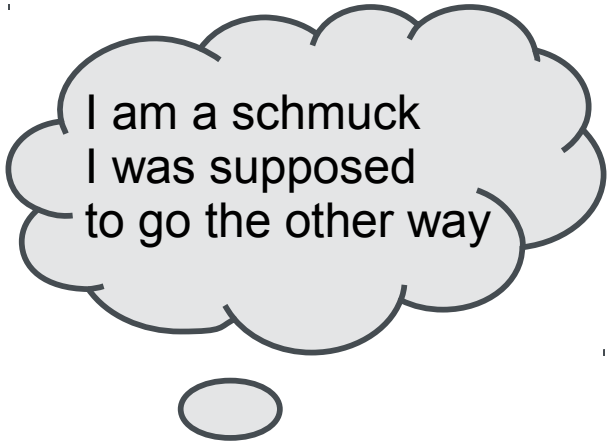
Thoughts about updates

- Lets say there is a security update, habitual practice
  - Apply update
  - Restart service

Incorrect, sorry play again

What happened?

- You just shut down your docker container
- On restart the original container with the vulnerability starts
- How did we get here?



It's called the human condition

Show of hands

The human condition

- For ages:
  - Apply security fix, restart service
- Containers
  - Rebuild from scratch
  - Apply security fix, commit, restart the container

Bet ya

the human condition will keep plenty of vulnerable containers running

For production then

- A re-training of habits is necessary
  - Not a small task

But....

- How did we know the container needed a fix in the first place?

Back to the beginning

- The vendor shipped us a new container
  - Do we want to trust an ISV to provide fixes in a timely fashion when they don't want to be in this business in the first place?
  - Does the integrator have the expertise and people on the embargo lists to react in a timely fashion?
- Built the container yourself?
  - Have a tracking mechanism for all the stuff that is in your containers?

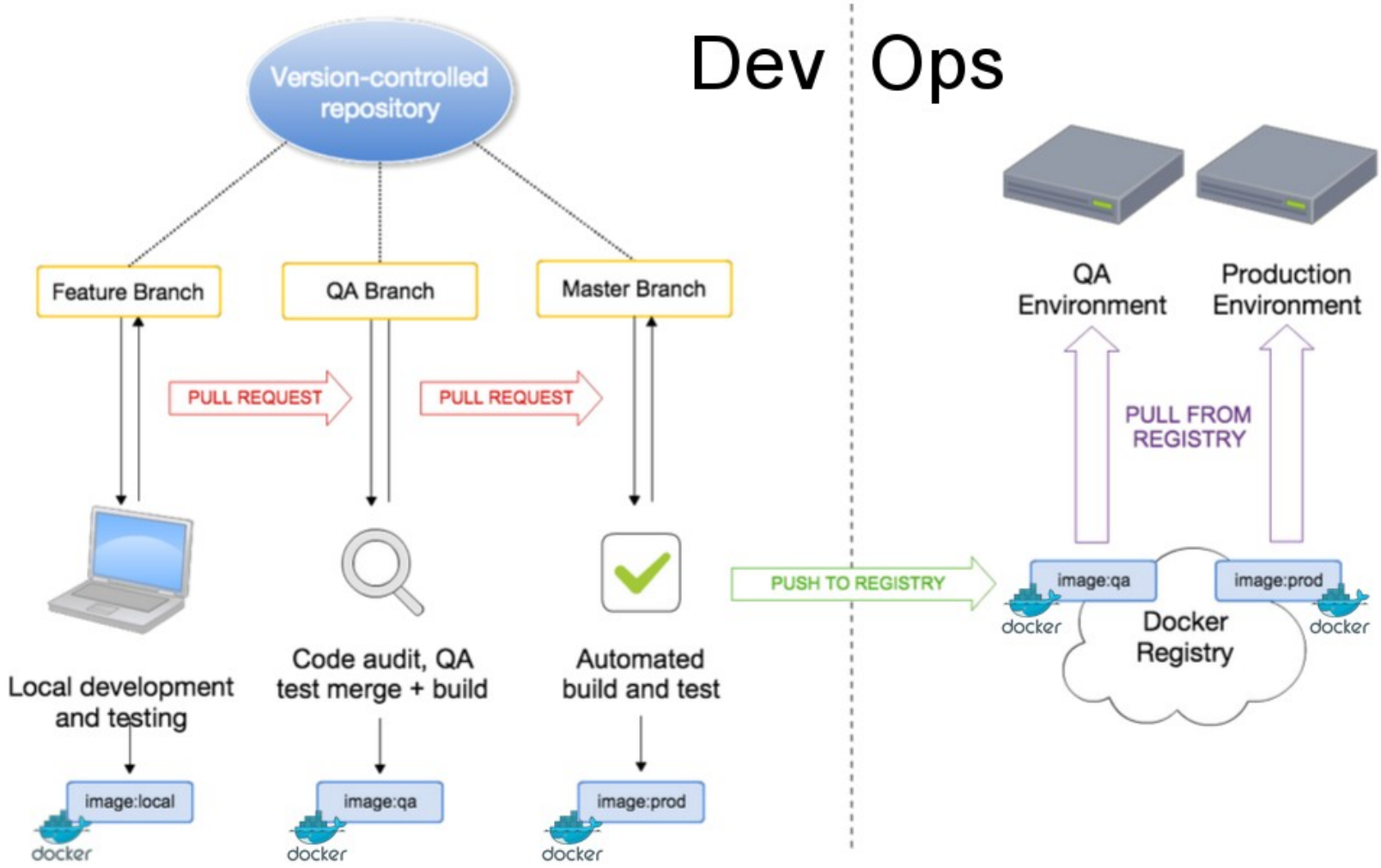
Problems solved?

- Application portability?
  - Only partially
    - ISVs generally don't want to ship containers
    - Container builder still has to worry about the “right” library versions
    - What about those applications that are kernel dependent?

What does it all mean?

- For production environments
  - Problems are just being pushed around, no solution
    - Spare a small subset of applications
  - There are no general solutions yet
  - A lot of work is left to do
- But containers do have their place
  - New applications developed with containers in mind
    - Although the whole “container content problem” remains

Not all doom and gloom – Novacoast



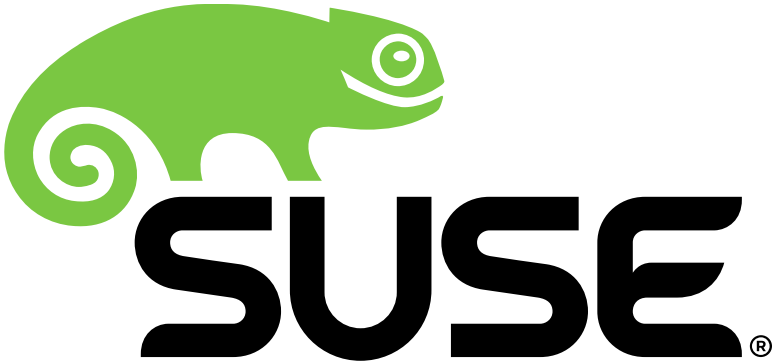
Summary

- Containers do solve problems in the right circumstances
- But be aware of the undifferentiated “it's a silver bullet” hype


DANKSCHEEN
BİYAN
SHUKRIA
YAQHANYELAY
TASHAKKUR ATU
GRACIAS
SUKSAMA
THANK
ARIGATO
SHUKURIA
GRAZIE
MEHRBANI
YOU
BOLZÏN
MERCI
JUSPAXAR


THANK
YOU!

Questions?


SUSE®
We adapt. You succeed.