Working with Multi-architecture Containers in Jenkins

Chris B. Behrens
SOFTWARE ARCHITECT

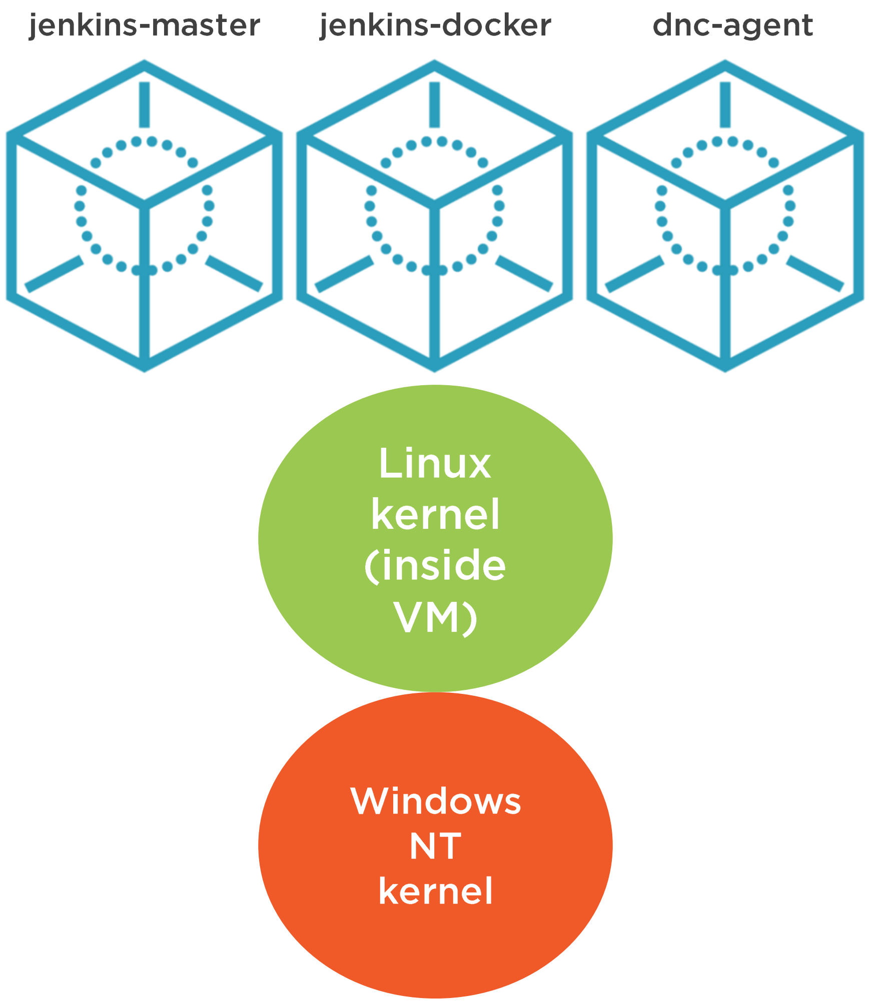
@chrisbbehrens

Understanding Multi-architecture

Using Jenkins and Docker to get typical work done

Now we’re shifting focus to creating Docker images

Docker Architecture



Separate Images for Separate Cases

Different Oses, different images

- ourenterprise/ourproduct-windows
- ourenterprise/ourproduct-linux

BusyBox

- Runs on different POSIX compliant architectures
- Using a multi-architecture repository

Dot Net Core Supported Architectures

Linux

OS	Version	Architecture
Red Hat Enterprise Linux	6+	x64
Red Hat Enterprise Linux	7, 8	x64
CentOS	7, 8	x64
Oracle Linux	7, 8	x64
Fedora	30+	x64
Debian	9+	x64, ARM32, ARM64
Ubuntu	16.04+	x64, ARM32, ARM64
Linux Mint	18+	x64
openSUSE	15+	x64
SUSE Enterprise Linux (SLES)	12 SP2+	x64
Alpine Linux	3.8+	x64, ARM64

Windows

OS	Version	Architecture
Windows Client	7 SP1+, 8.1	x64, x86
Windows 10 Client	Version 1607+	x64, x86
Nano Server	Version 1803+	x64, ARM32
Windows Server	2012 R2+	x64, x86

macOS

OS	Version	Architecture
Mac OS X	10.13+	x64

Why This Matters

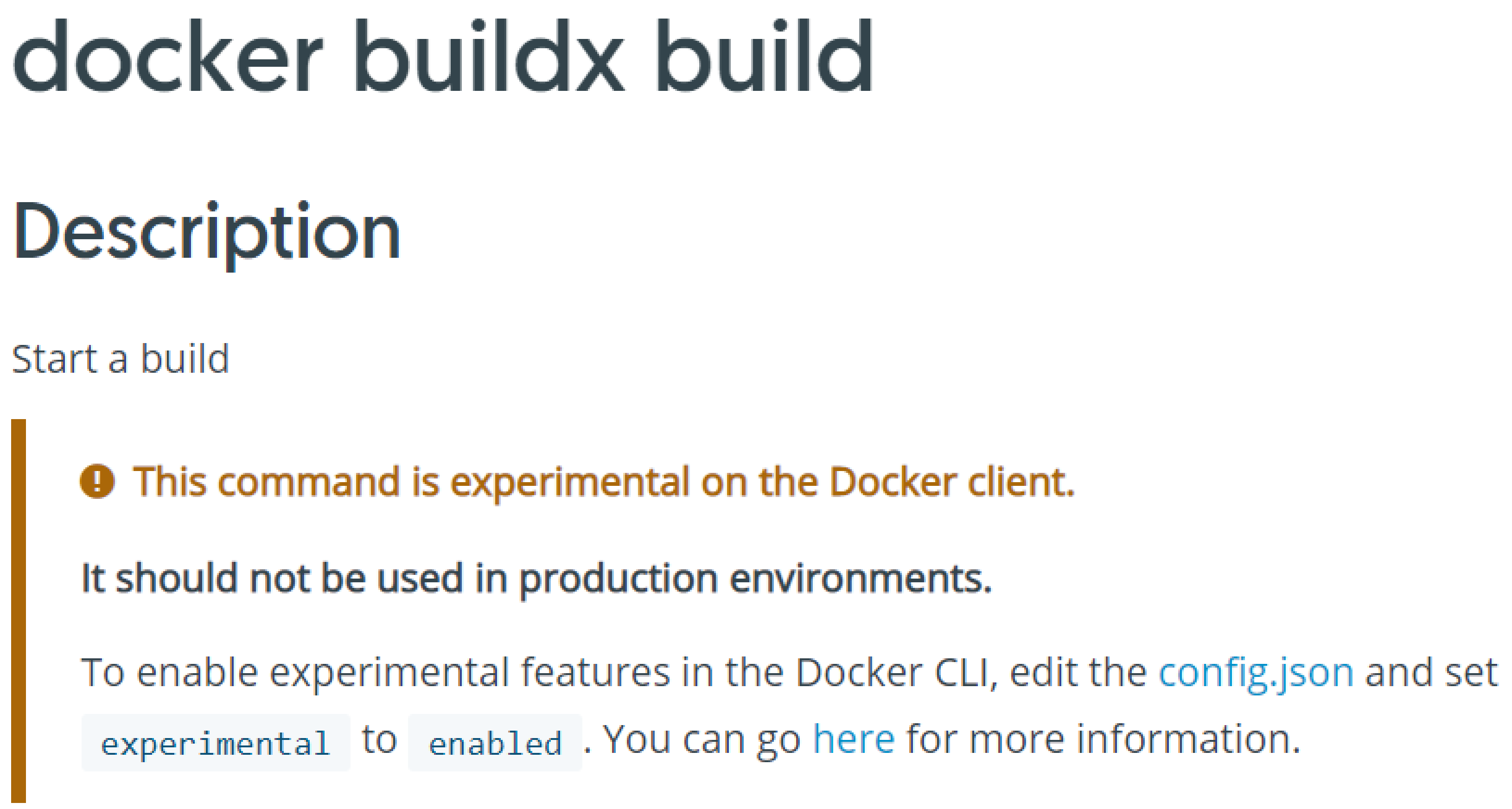
Who cares? Everybody has multiple installers

At least a virtual machine for a different OS

Raspberry Pi – running on an ARM processor architecture

And Now, a Warning

And Now, a Warning



docker buildx build

Description

Start a build

> ● **This command is experimental on the Docker client.**
>
> **It should not be used in production environments.**
>
> To enable experimental features in the Docker CLI, edit the config.json and set `experimental` to `enabled`. You can go here for more information.

Building Your Docker Images for Multi-arch

The manifest

An asset for negotiating platform and architecture

Demo

Look at a Windows platform image in DockerHub

Try to pull it locally

Fail

Break down what’s going on

How BuildX Builds for Platforms You Don’t Have

We don’t have ARM

But we do have an ARM emulator

QEMU – a hardware virtualizer

MAME – the Multi-Arcade Machine Emulator

In the eighties, arcades were king

To make a video game today, you stick a PC and a monitor inside a cabinet

But back then, arcade games could be entirely different from each other

- Including different processors

A hardware emulation layer

- Which can load the original arcade game code
- So that the behavior of the game is identical (mostly) to the original

QEMU is MAME for processor architectures

Demo

Create a minimum viable understanding of Multi-arch

By looking at a simple Dockerfile

Using Buildx to build our image for both architectures

- For AMD64 processors
- For ARM

Building a manifest for our Dockerfiles

Pushing it to DockerHub

Testing it out locally

Build with BuildKit

A new engine for building Docker images

Executes image steps in parallel

BuildKit *itself* is stable

Demo

Modify our Jenkins Docker agent image so that the experimental features are enabled

Create a Jenkinsfile that executes our buildx command from before

Get that into version control

Making This Work in Jenkins

Take our agentdnc image

Add a single element

- The environment variable to enable experimental options (like BuildX)

Making This Work in Jenkins

1. Create a custom build space, cbbspace
2. Tell docker to use that builder
3. Copy our Dockerfile to container
4. Create a credential set for docker to use to authenticate to DockerHub
5. Execute our multi-arch build and push up the images

Demo

Script our build process

Restrict that build to our new experimental agent

Configure that agent in Jenkins

Execute a build

Verify the results in DockerHub

Multi-arch Wrap-up

https://hub.docker.com/_/microsoft-dotnet-core-samples

We kept it simple (believe it or not)

Multi-stage Jenkinfiles as well as multi-stage Dockerfiles

Build our app on the SDK image, deploy it to the Runtime, just like the samples

But this oversimplifies the build

https://github.com/dotnet/dotnet-docker/blob/master/samples/dotnetapp/Dockerfile

Making This Work

1. Execute the build sequence in the pipeline

- SCM
- Build
- Test
- Package

2. Move the output of the build into the image

3. Orchestrate containers? Maybe

Summary

Took the time to understand the multi-architecture challenge

The traditional way to tackle this

How BuildX solves this problem

- Hardware emulation with QEMU
- The experimental features of Docker

Manual demo of a multi-architecture build

A Docker in Docker experimental multi-architecture build