

# **CAB432 Cloud Computing**

## **Lecture 2: Container as a Service**

Faculty of Science





The basics of being a container

# DOCKER OVERVIEW



a university for the **real** world<sup>®</sup>

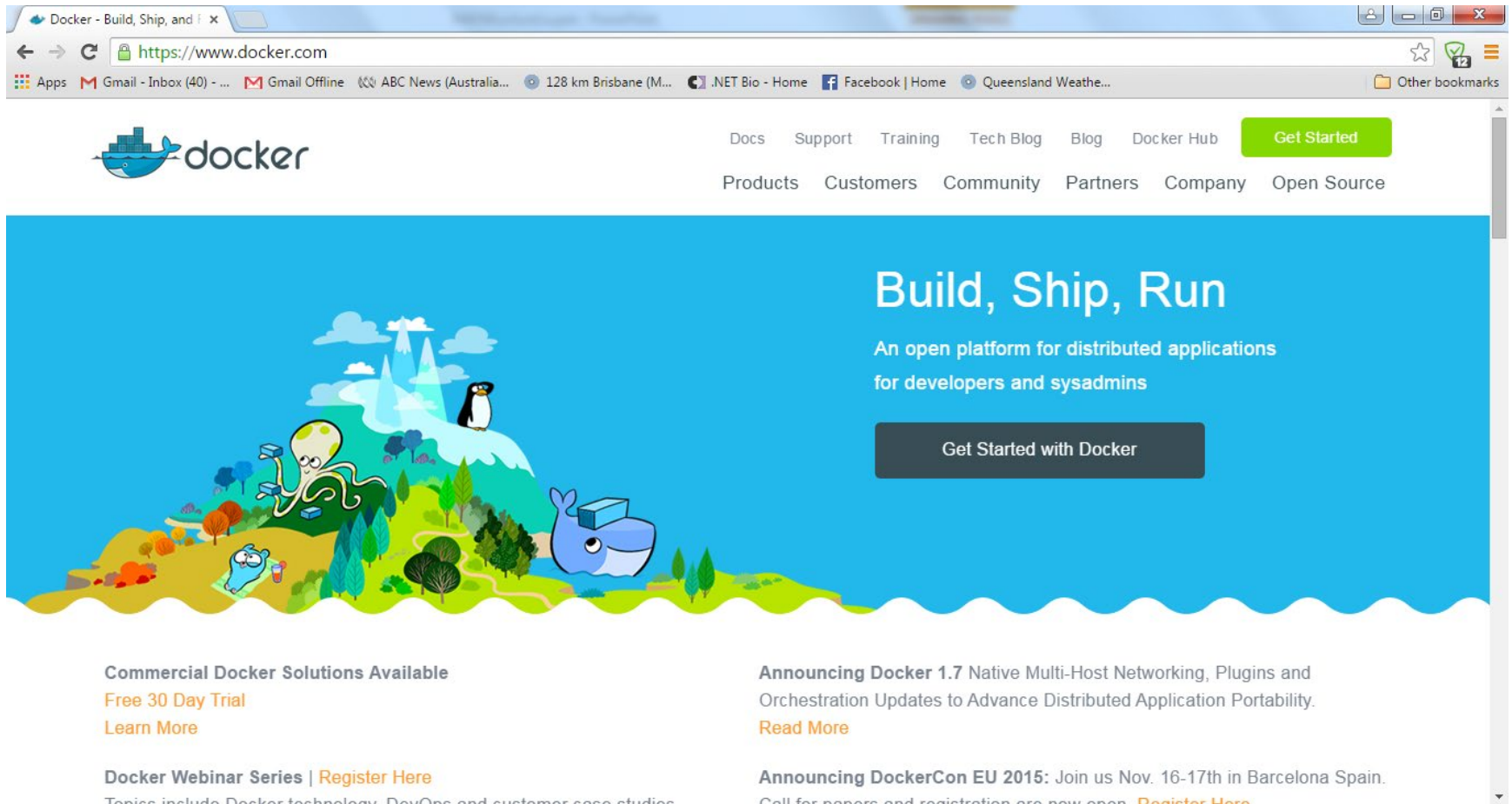
# Cloud = Utility Computing + Elasticity + Scale

- Utility Computing means virtualization
- IaaS virtualizes an entire “stack”
  - Hardware, operating system/device drivers
  - Optionally: middleware [databases, app server, messaging, etc.]
  - Some Applications
- High setup, maintenance cost
  - Hypervisors and VM Monitors
  - Entire stack even when we don't need it
  - Consumes excess resources – storage, CPU
-

# Container as a Service (CaaS)

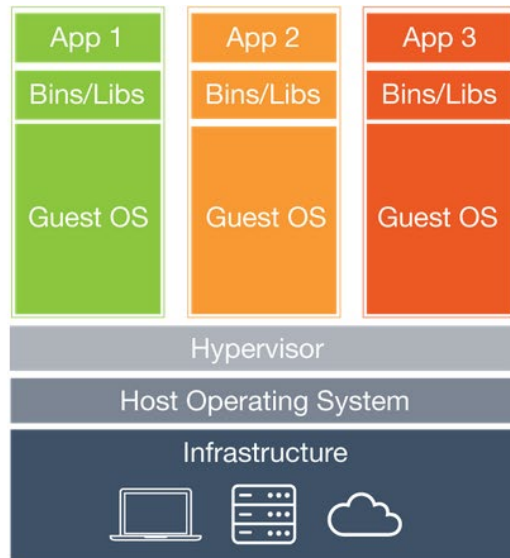
- Alternative: Don't virtualize the whole machine
- Focus on a custom set of services:
  - Basic OS, middleware services and applications
- Main Example: Docker (<https://www.docker.com>)
  - Many other alternatives
- Orchestration:
  - Kubernetes - <https://kubernetes.io/>
  - Docker Swarm - <https://docs.docker.com/engine/swarm/>
  - All the majors now have managed container services

# Docker

A screenshot of the Docker website homepage as it appeared in 2016. The browser window shows the URL https://www.docker.com. The page features the Docker logo (a blue whale) and a navigation menu with links to Docs, Support, Training, Tech Blog, Blog, Docker Hub, Products, Customers, Community, Partners, Company, and Open Source. A prominent green 'Get Started' button is in the top right. The main banner has a blue background with a colorful illustration of a penguin, an octopus, and a whale on a small island. The text 'Build, Ship, Run' is displayed in large white letters, followed by 'An open platform for distributed applications for developers and sysadmins'. A dark grey button with the text 'Get Started with Docker' is centered below the banner. Below the banner, there are four sections of text: 'Commercial Docker Solutions Available' with a 'Free 30 Day Trial' and 'Learn More' link; 'Announcing Docker 1.7' with details about native multi-host networking and a 'Read More' link; 'Announcing DockerCon EU 2015' with details about the event and a 'Register Here' link; and 'Docker Webinar Series' with a 'Register Here' link and a list of topics.

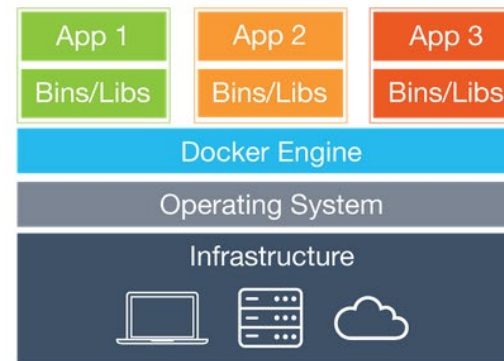
The 2016 slide, but so much more interesting than the current page 😊

# Simplified Virtualisation



## Virtual Machines

Each virtual machines includes the application, the necessary binaries and libraries and an entire guest operating system - all of which may be tens of GBs in size.



## Containers

Containers include the application and all of its dependencies, but share the kernel with other containers. They run as an isolated process in userspace on the host operating system. They're also not tied to any specific infrastructure – Docker containers run on any computer, on any infrastructure and in any cloud.

# Application Virtualisation

- Layered **on top** of an OS (typically, Linux)
  - Uses OS capabilities (Linux cgroups)
- Does **not** virtualise everything
  - hardware + firmware + OS + middleware + standard tools
- Precise, user-defined set of software and configuration
  - No need to clone the VM – so much lighter and faster
  - Script instantiates a cloned container in a matter of seconds (your mileage may vary)
  - Pre-canned scripts or images (see next slides) may be shared with others

# Pros and Cons

- Pros:
  - *Much* more efficient use of “bare metal” hardware resources
  - Orders of magnitude (multiple digits) more containers than VMs
  - More efficient use of CPU cycles, main memory
  - No need to maintain an entire OS stack
  - Simpler management and orchestration c.f. Puppet and Chef
  - Reboot times orders of magnitude better than rebooting VMs



# Pros and Cons

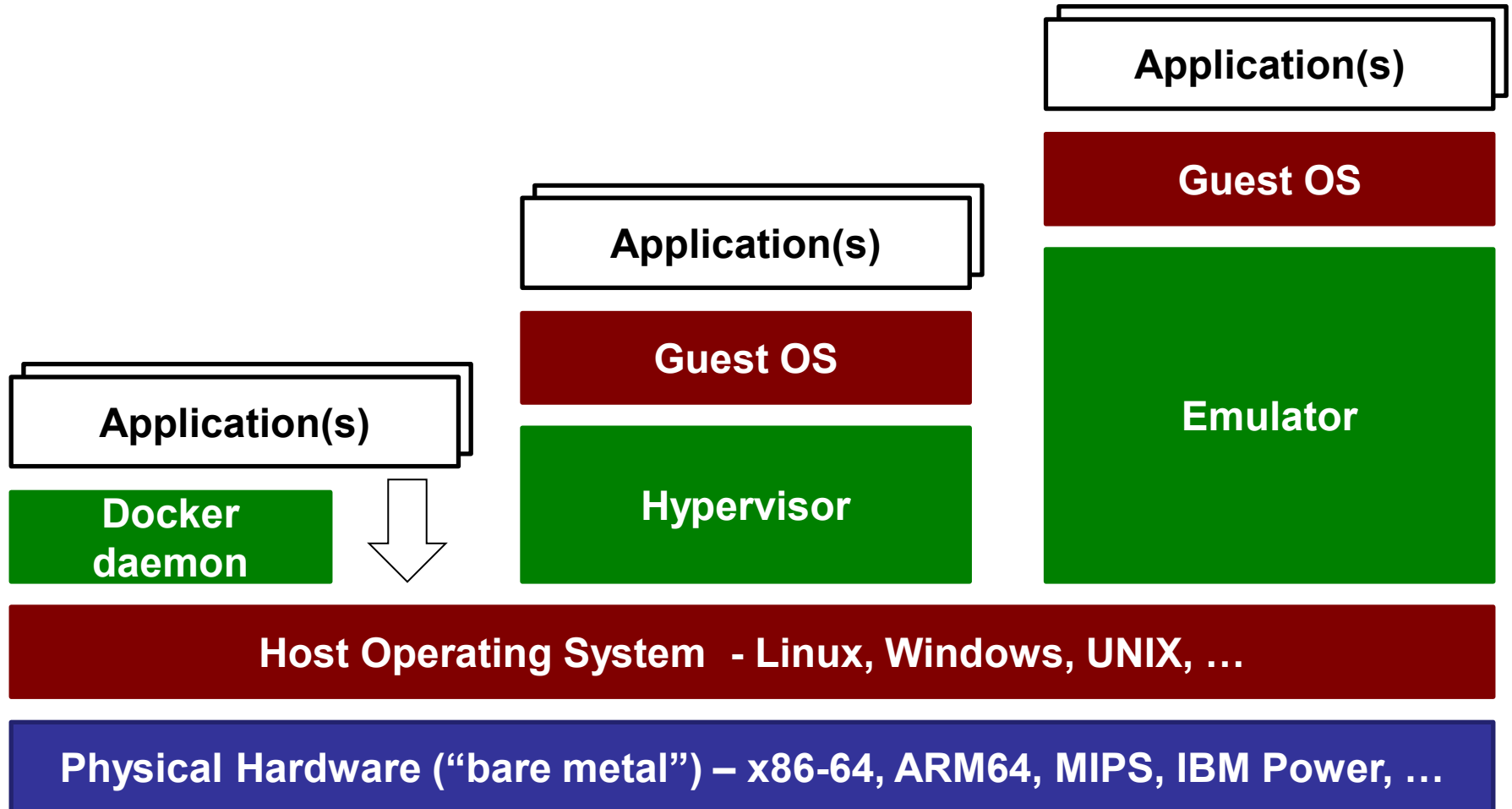
- Cons:
  - Assumes guest software to be compatible with Linux
  - True VMs allow better isolation
  - Containers have possible port collisions
  - No access to OS-level daemons

# CaaS vs Virtualization vs Emulators

CaaS

Virtualization

Emulator



# Next Steps

- We will learn about virtualisation
- Then we will learn to use Docker
- Finally (later) we will learn about Kubernetes & Swarm