The Docker Revolution

by @twombh February 2017

Vertical versus Horizontal

How to find a web page

Vertically

Step 1: Get a very, very, very big computer.

Step 2: Get a very, very, very fast Internet connection.

Step 3: Load every page on the Internet.

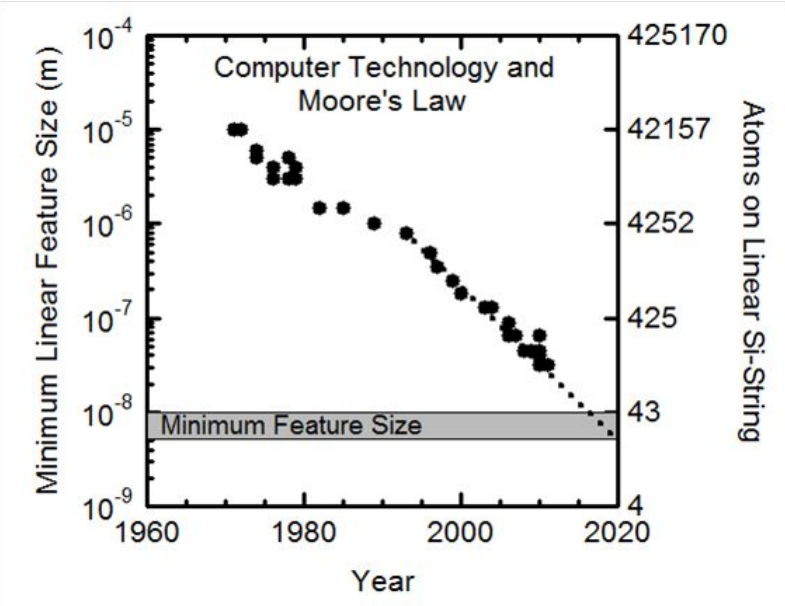
Horizontally

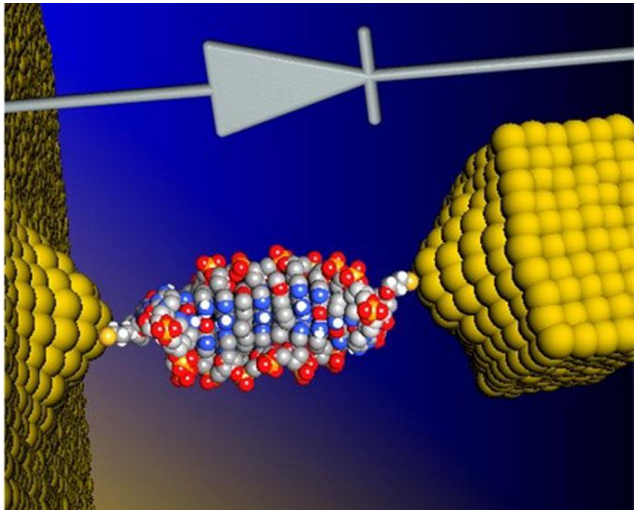
Step 1: Get lots of computers to scrape the Internet. Yesterday.

Step 2: Store just the text using something like Balanced Binary Trees.

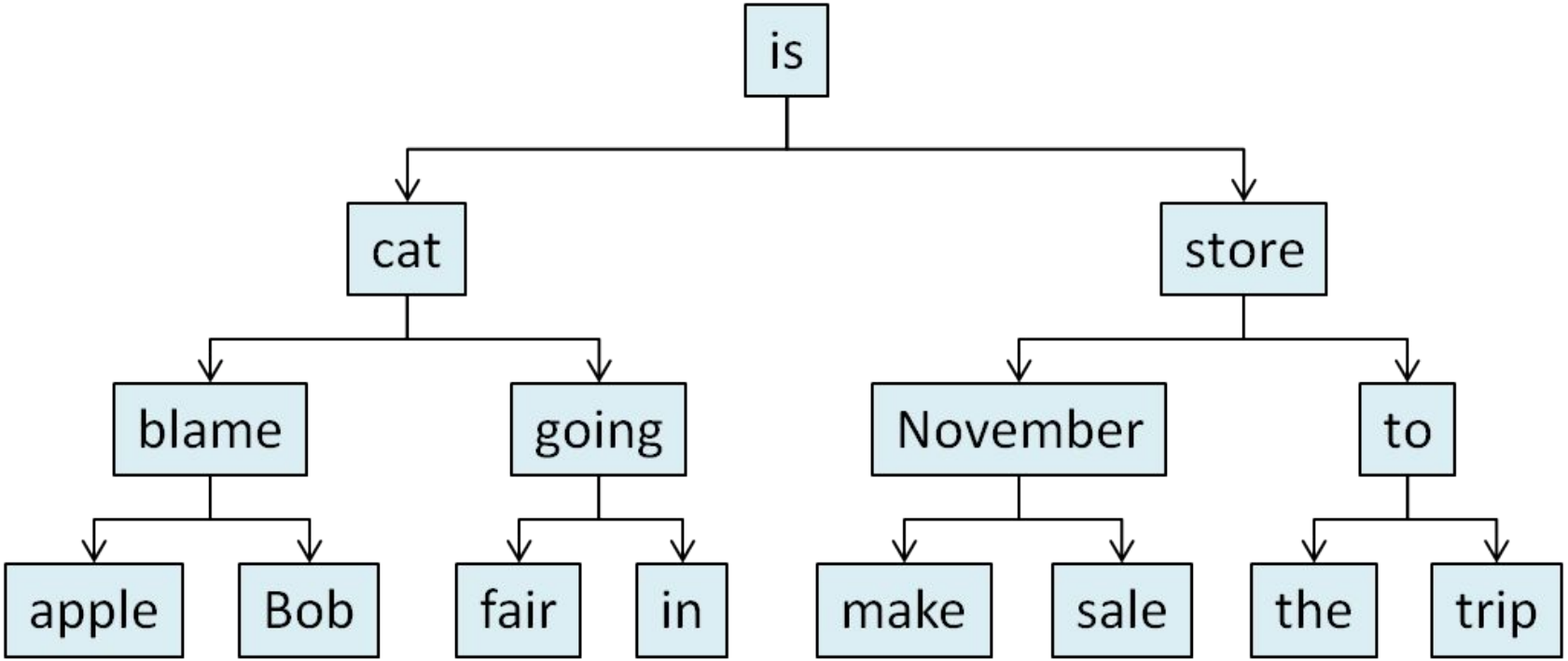
Step 3: Search the Internet in RAM.

The limits of physics





Balanced Binary Search Tree



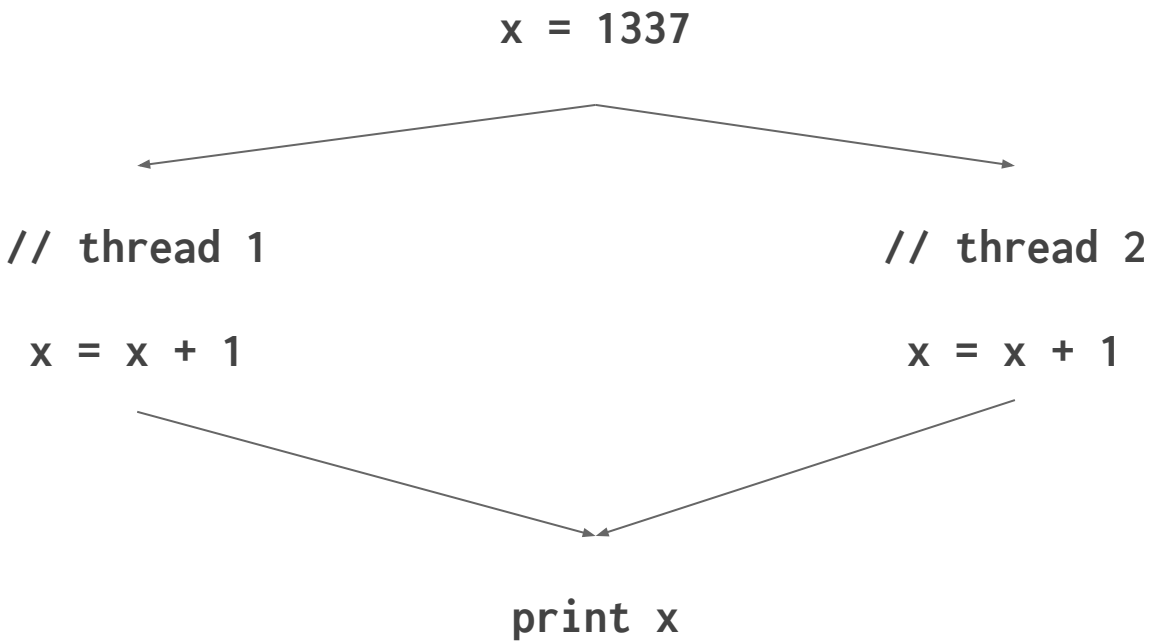
Concurrency versus Parallelism

Interruptible versus Independent

Walking on the street is **interruptible.**
But answering your phone is **not independent.**

Putting out a fire is **independent.**
But **not interruptible.**

Race Conditions



Handling lots of web requests

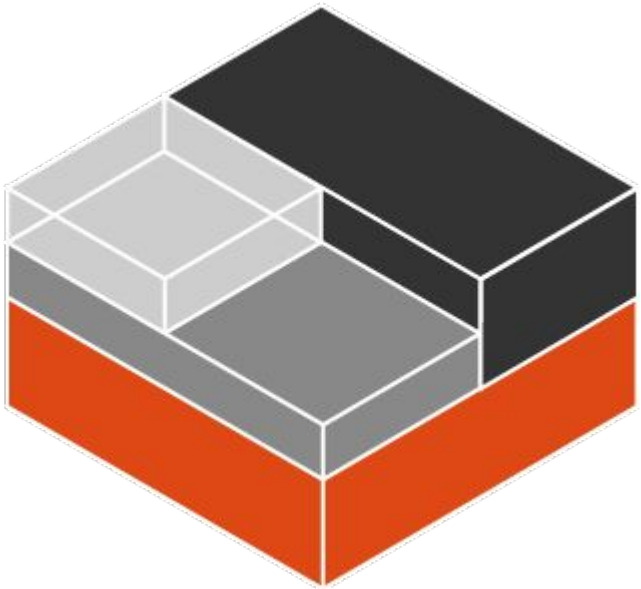
* Independent, but not interruptible. Eg; failed email delivery messages
* Racks with 256RAM and 16 cores :)
* VMs are programmable, which saves labour and money.
* **Arbitrary scaling** and parallelism mean we can forget about physics.
* ‘Elastic computing’ saves money.

Independence As A Service

* Heroku was a natural evolution of shared hosting. But not using VMs.
* Less access to servers (no FTP, no permanent storage, etc) and strict requirements (git, all external services must be run separately).
* But it could be easily and arbitrarily scaled.

Formalisation of Parallelisation

* LXC: Linux containers.
* Google and other large compute companies.
* Doesn’t virtualise CPU/RAM it restricts access to them.
* More like `init` process than a VM.
* PHP, Python, Ruby, Java: all run on Linux.



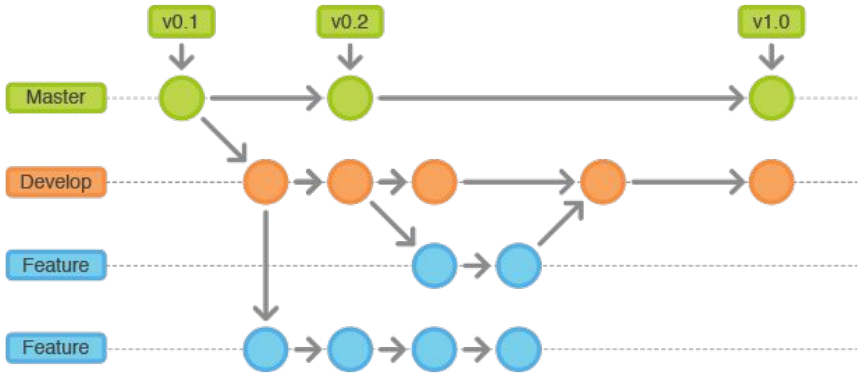
Solomon Hykes and dotCloud

* dotCloud were essentially a Heroku competitor.
* Hykes was using LXC.
* Docker was borne from the tools Hykes was building at dotCloud.
* March 2013.
* He was simply open sourcing a way of parallelisation that had been around for a few years.

The 12 Factor App Manifesto

I. Codebase One codebase tracked in revision control, many deploys.
II. Dependencies Explicitly declare and isolate dependencies.
III. Config Store config in the environment.
IV. Backing services Treat backing services as attached resources.
V. Build, release, run Strictly separate build and run stages.
VI. Processes Execute the app as one or more stateless processes.
VII. Port binding Export services via port binding.
VIII. Concurrency Scale out via the process model.
IX. Disposability Maximize robustness with fast startup and graceful shutdown.
X. Dev/prod parity Keep development, staging, and production as similar as possible.
XI. Logs Treat logs as event streams.
XII. Admin processes Run admin/management tasks as one-off processes.

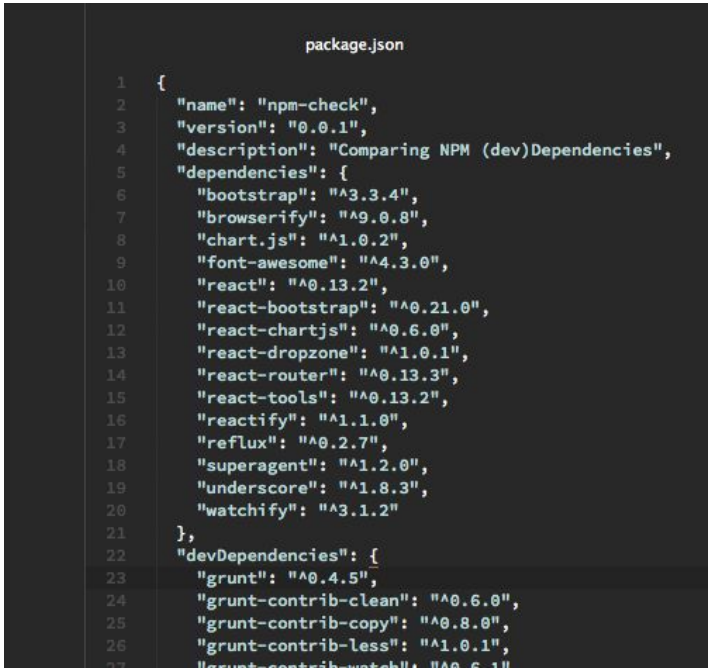
Codebase in Revision Control



* Backup, querying history, collaborating (like a mutex).
* Moving code between live, distributed, concurrent, containers is quicker.
* Prevents hotfixing culture.
* Better for onboarding new devs.
* `` `docker pull/push` ``.

Explicitly Declare Dependencies

* Code does not exist in isolation. Depends on DB protocol versions etc.
* Independence must carefully acknowledge dependence.
* `docker build`

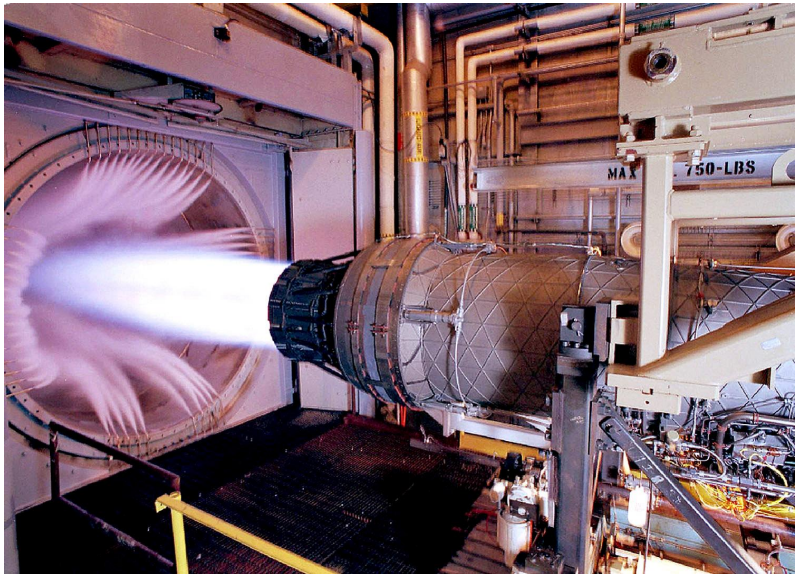


Store Config in the Environment

* Better for testing. Forces you to decouple code.
* Single Responsibility Principle.
* The 4 cornerstone environments: developers/testing/workers/production.
* Hot-swapping dead databases.
* `docker run --env`

Testing Gives You Half of Docker

* Dependencies, Config, Backing Services, Seperated Build/Run, Disposability.
* Testing requires a fundamental shift in how you write code.
* Similar to Docker, it requires changing fundamentally changing your workflow and the way you think.



The Utopian Vision

* Docker isn't just a thing to put apps in.
* It aims to improve the entire world of creating, testing, collaborating, deploying, updating and scaling apps.

Democratisation of the Internet

* If an app can be completely made independent then there is no need for the company surrounding the app.
* Apps can be deployed directly from Github with no devops.
* Imagine if all the essential Internet services such as search and social media were funded like Wikipedia.

Danabox

Where to Start?

Checkout **Awesome Docker on** Github

https://github.com/veggiemonk/awesome-docker