Deployment

Within a Traditional IT environment

Rob Allen

akrabat.com ~ @akrabat ~ January 2017

What?

Software deployment is all of the activities that make a software system available for use.

Wikipedia

Traditional IT environment

for Business

Windows Server

Not in this talk

- Cloud deployments
- Provisioning of servers
- Continuous Delivery
- Chat Ops

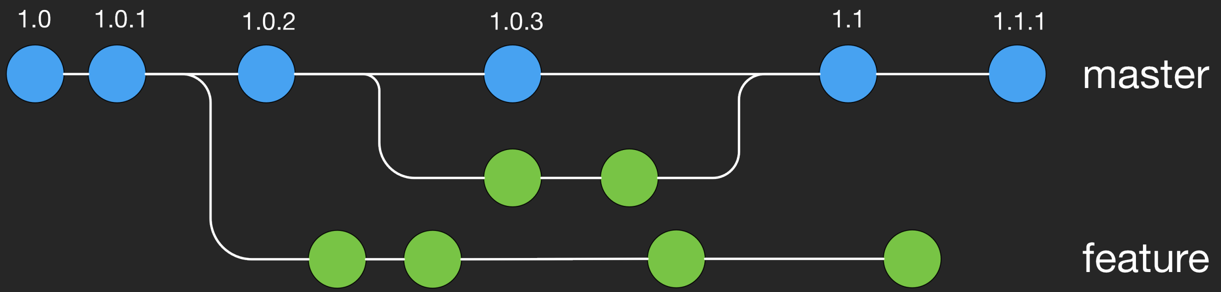
Fundamentals

Development processes

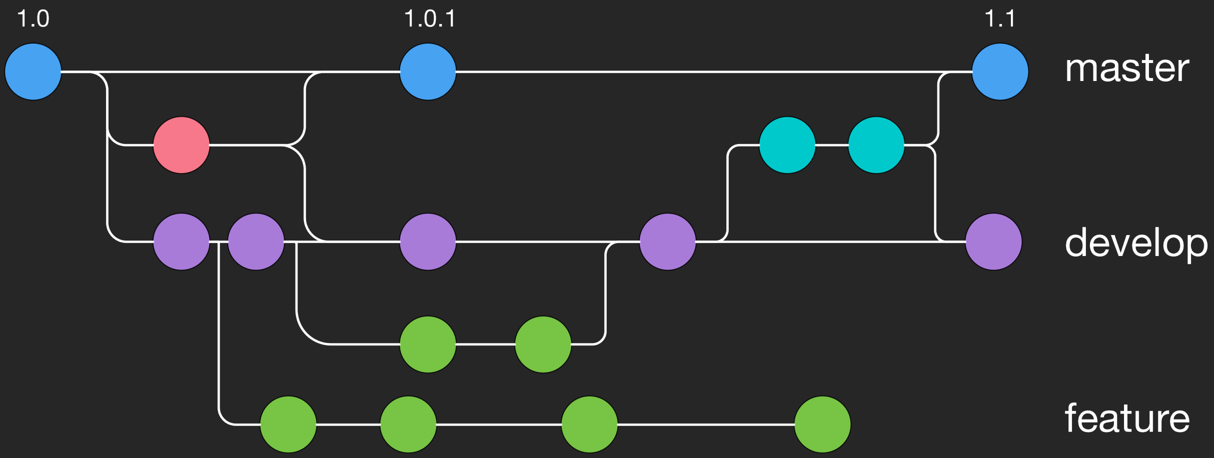
Code organisation

- Isolate work
- Work on branches
- Always have a way to bugfix live

Feature branches



Git flow



Automatic testing

Automatically test every PR

- Coding style
- Unit tests
- Functional tests
- Acceptance tests

DB schema management

Migrations

- Schema changes stored in individual files
- *delta* files with UP and DOWN functionality
- It really *doesn't matter* which tool you use

Configuration data

Context awareness

Each instance needs different configuration

- Environment variables (12factor.net/config)
- Local configuration file

Deployment

Manual deployment

Simply copy the files to the right place
(S)FTP / rsync / Remote Desktop

Manual deployment

- Slow
- Inconsistencies between each deploy
- Failures in production
- No records. Who deployed what, when (or where)?

It's more than just code

Other things to think about

- File permissions
- Preserve user uploaded files
- After upload
  - Stale cache?
  - Cache priming?
- Keeping records

People aren't machines!

Automation

People:

good at creative judgement

Machines:

good at doing the same thing over and over

Benefits

- Deferred deployments
- Self-service deployments
- Automatically triggered deployments
- Smoke tests
- Automatic rollbacks

Scripts

- Provide consistency
- Provide a platform for additional abilities
- No need to wait for other people
- Faster and less hassle.

Deployment checklist

- Copy files & folders
- Database schema & content updates
- Configuration updates
- File permissions
- Clear caches

Tooling

- Simple script
- Make-like tool
- Specialist tool

Simple script

- Bash
- Powershell
- Usually runs on the server

Make-like tool

- Abstracted script
- Easier to make cross platform
- Make/Phing/Ant/Rake
- More likely to push to server

Specialist tools

- Fabric/Capistrano
- Ansible/Chef/Puppet/Salt
- Deploybot
- More likely to be automated

Challenges

Technical

Infrastructure challenges

- Platform
- Permissions

Political

- What is acceptable?
- What are you allowed to do?
- Does IT change things for you?
- Does IT change things unexpectedly?
- Change request documentation?

Case studies

Case study 1: Internal

- Self-hosted by client
- Within company network
- Windows/IIS/SQL Server

Case study 1: Internal

Multi-script solution

1. Script to tag code & build tarball
2. Ship tarball to UAT server via VPN
3. IT copy tarball from UAT to Live
4. Script to untar & run deploy steps

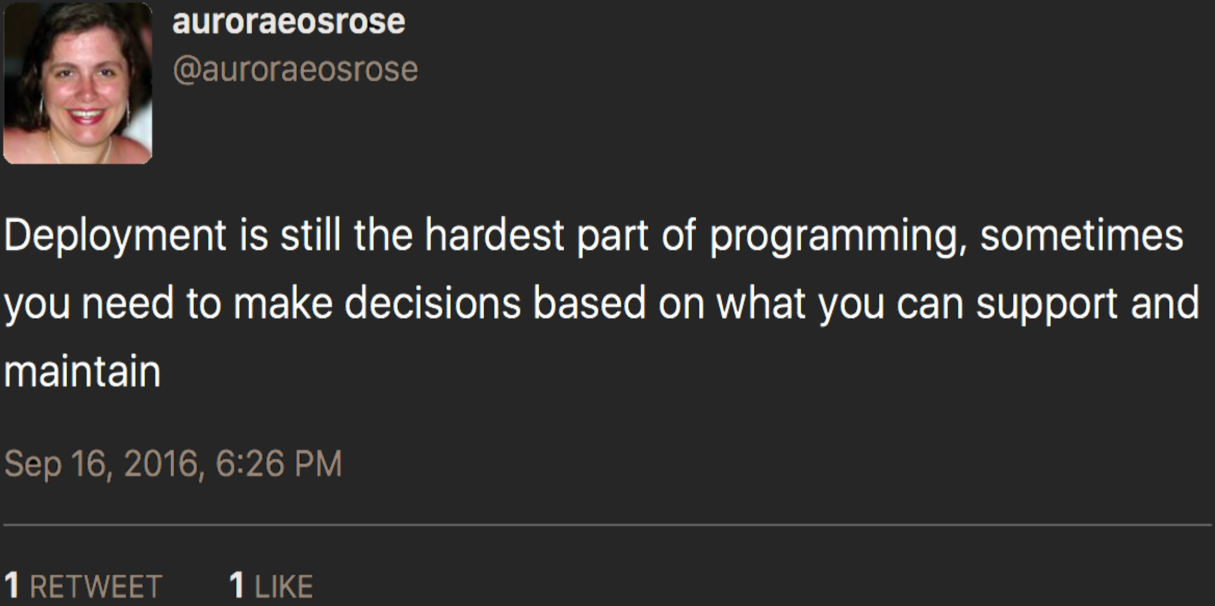
Case study 2: Minicomputer

- Self-hosted by client
- Some access to Internet
- IBM i-series/DB2

Case study 2: Minicomputer

- CI tooling
- Checklist
    - Manual tag creation for release
    - Automatic script to deploy

Final thought



auroraeosrose
@auroraeosrose

Deployment is still the hardest part of programming, sometimes you need to make decisions based on what you can support and maintain

Sep 16, 2016, 6:26 PM

1 RETWEET **1** LIKE

Thank you!

Rob Allen ~ akrabat.com ~ @akrabat