Deployment Tools

Ensembl Retreat
Dec 1-2 2016

EMBL – European Bioinformatics Institute
Wellcome Trust Genome Campus
Hinxton, Cambridge, CB10 1SD, UK

Goals

- Overview of (some) existing tools
- How deployment tools work
- How they improve speed/quality of software releases

How this session runs

- Short talks
  - Andy – *CPAN/DarkPAN*
  - Prem – *Docker*
  - Nick – *REX*
  - Thibaut – *Homebrew*
- Free discussion
  - get inspired by topics/ideas

    https://www.ebi.ac.uk/seqdb/confluence/display/ENS/Deployment+tools+2016#Deploymenttools2016-Topics/Ideas

Deployment

"*grouping of every activity that makes a program available for use and moving it to the target environment*"

- Process with several interrelated activities, transitions
  - occur at the producer or consumer side
- precise processes cannot be defined
  - every software system is unique

Deployment Tools

- Any software instrument/platform supporting the deployment process
- Large ecosystem
- Focus: automating the **deployment pipeline**

Continuous Delivery

“Software production process where software can be released to production at any time with as much automation as possible at each step”

- advocates creation of automated deployment pipeline
- release software rapidly/reliably

Deployment Pipeline

- Build/Deploy/Test/Release cycle
  - automated implementation
- Enables self-served releases of any application in any environment
- Optimise cycle time: avg time between prod releases
  - dev cost are lowered
  - release failure risk minimised
  - faster customer feedback loops

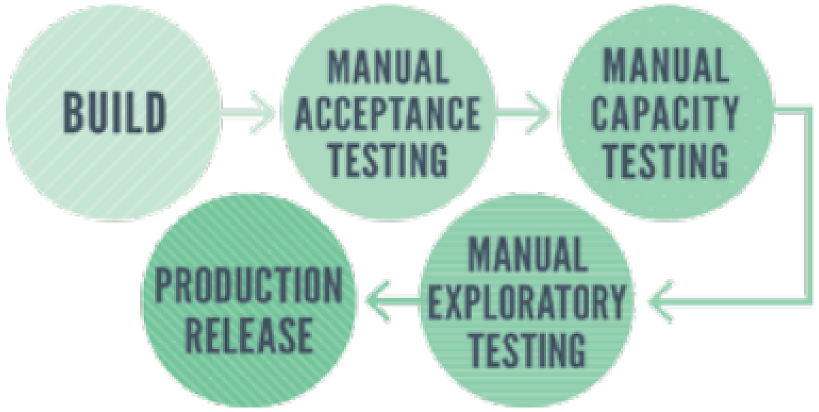
CD: where it's coming from

- Continuous Integration (CI)

  "*development process where a (CI) server rebuilds a branch of source code every time code is committed*"

  – extended to include deployment/installation/testing into prod
- focused on development

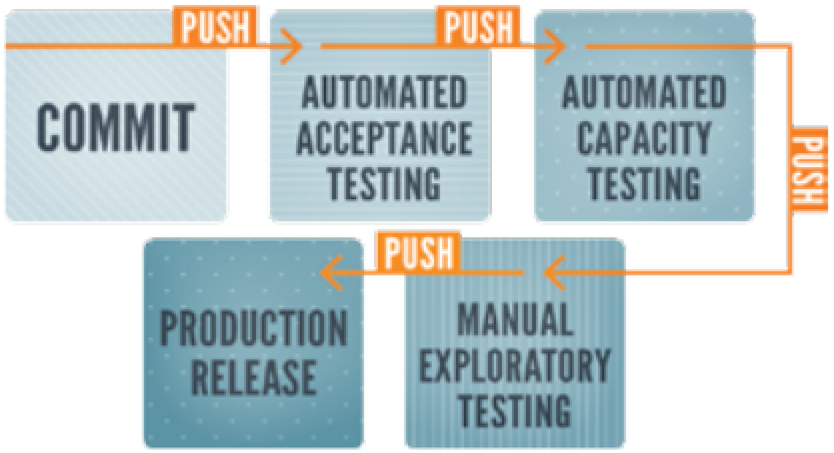
  – benefit fraction of release process



Source: DZone

Beyond CI

- CD: extension of CI practices into infrastructure mgt and prod env
- Infrastructure as code
  - version control, automated testing, deploy tools


COMMIT
PUSH
AUTOMATED ACCEPTANCE TESTING
PUSH
AUTOMATED CAPACITY TESTING
PUSH
MANUAL EXPLORATORY TESTING
PUSH
PRODUCTION RELEASE

CD Toolchain

- No single tool, automation product or deployment pipeline impl provides CD
- CD impossible wo some capabilities the tools provide
- Tool categories:
  - Orchestration and deployment pipeline visualisation
  - Version Control
  - CI
  - Artifact Management
  - Test and Environment Automation
  - Server Configuration and Deployment
  - Monitoring and Reporting

Orchestration & Visualisation

- Backbone of CD
  - allow building effective sequence of steps
  - provide visualisation utilities
  - detect and expose delays at each stage
- Can use dedicated deployment pipeline tools or Application Release Automation (ARA) solution

Orchestration and Visualisation

Deployment pipeline tools

- Jenkins
- Travis CI
- Thoughtworks GO
- Atlassian Bamboo
- ...

ARA

- ElectricCommander
- IBM UrbanCode, XebiaLabsXL
- CA Lisa

Version Control

- All text-based assets should be stored in a version control system
  - easily accessible by anyone
  - code changes very easy to review
  - configuration files defining build/release system
- Git, Subversion, Mercurial, Perforce, TFS

Continuous Integration

- Can support orchestration/visualisation
- Core functionality
  - integrate new code into stable main release
  - alert if issues with new code
- Team should also connect a code metrics and profiling utility
  - stop integration if metrics reach a threshold

CI (continued)

- CI tools: Jenkins, TravisCI, ThoughtWorks GO, CircleCI, Jetbrains TeamCity, Atlassian Bamboo
- Code Metrics: SonarQube, SLOC, SciTools Undestand

Artifact Management

- Artifacts: assembled pieces of an application
  - application code and assets
  - infrastructure code
  - VM images
  - configuration data
- Packaged artifacts (not source code) are focus of deploy pipelines
- Metadata identifies when/how package was tested/deployed in an environment

Artifact Management

- Artifacts should be traceable
  - Identifiable (unique name)
  - Versioned (semantic)
  - Immutable
- Management is done with Artifact Repository Manager
- Art Repositories contains complete usage history with dependency resolution
  - Can track exactly what was (or will be) tested/deployed

Artifact Management

- OS-level package managers
  - APT, RPM, **Homebrew**
- Language-specific package manager
  - **CPAN,** PIP, Ruby Gems, Composer
- Repository Managers
  - Archiva, Artifactory, Nexus

Test/Environment Automation

- All tests should be automated, except:
  - Exploratory testing, UI design ispections, UATs
- Automated testing tools should be lightweight and operate non interactively
- Teams create testing environment on-demand using env automation tools
  - provision VM and configure environment template

Test/Environement Automation

- Test automation: JMeter, Selenium/WebDriver, Cucumber, RSpec, LoadUI, PageSpeed, Netem, SoapUI, Test Kitchen
- Environment automation:
  - Vagrant, **Docker**, Packer

Server Configuration/Deployment

- Distribution/Installation of packages
- Two main deployment models
  - **Push**: Capistrano (Ruby), Fabric (Python), **REX** (Perl), ThoughtWorks GO, various CI/build/ARA tools
  - **Pull**: Ansible, Chef, CFEngine, Puppet, Salt

Push Model

- Master server manages distribution/installation of packages to multiple remote machines
- Pros: (good choice for small systems)
  - simplicity: easy to set up and run
  - control: everything is synchronous
- Cons:
  - lack of full automation: server does not boot and configure itself
  - not scalable

Pull Model

- AKA: configuration management systems
  - server acts as master
  - clients pull config information from master and figure out what to do
- Pros:
  - full automation capabilities
  - increased scalability: clients contact server independently
- Cons:
  - proprietary conf mgt language (ex. Chef)
  - scalability still an issue, unless deploy several master servers

Monitoring/Reporting

- Essential for spotting pbs and halting pipeline
- Do not manually collect logs
- Logs should be shipped to and indexed in a central store
- Log store should be connected to all environments (incl. developer’s system)
  - speed up diagnosis and resolution

Monitoring/Reporting

- Log Aggregation & Search:
  - Fluentd, Graylog2, LogStash, nxlog, Splunk
- Metric, Monitoring, Audit:
  - Collectd, Ganglia, Graphite, Icinga, Sensu, ScriptRock