Introduction to DevOps

Omid Vahdaty, DevOps Ninja

© 2016 Omid Vahdaty, All Rights Reserved.

PERIODIC TABLE OF DEVOPS TOOLS (V1)

XebiaLabs
Deliver Faster

Os	Open Source
Fr	Free
Fm	Freemium
Pd	Paid
En	Enterprise

Database	SCM	Build
CI	Repo Mgmt	Testing
Deployment	Config / Provisioning	Containerization
Cloud / Iaas / Paas	Release Mgmt	Collaboration
BI / Monitoring	Logging	Security

1 En O 12c																			2 Fm Aws Amazon Web Services
3 Os My MySQL	4 Os Gt Git												5 En Ch Chef	6 En Pu Puppet	7 Os An Ansible	8 En Sl Salt	9 Os Dk Docker	10 Pd Az Azure	
11 En Mq MSSQL	12 Os Sv Subversion												13 Fr Ssh SSH	14 En Bl BladeLogic	15 Os Va Vagrant	16 Fr Tf Terraform	17 Os Rk rkt	18 Fm Hk Heroku	
19 Os Pq PostgreSQL	20 Fr Mc Mercurial	21 Os Mv Maven	22 Os Gr Gradle	23 En Mr Meister	24 Os Jn Jenkins	25 Pd Bb Bamboo	26 Os Tr Travis CI	27 Fr Ar Archiva	28 Os Fn Fitnesse	29 Fr Se Selenium	30 Os Gn Gatling		31 Pd Gd Deployment Manager	32 Os Sf SmartFrog	33 Fr Cb Cobbler	34 Os Bc Bcfg2	35 Os Kb Kubernetes	36 En Rs Rackspace	
37 Os Mg MongoDB	38 Fm Gh Github	39 Os Br Buildr	40 Os At ANT	41 Fm Bm BuildMaster	42 Fm Cs Codeship	43 Fm Sn Snap CI	44 Fm Cr CircleCI	45 Os Nx Nexus	46 Fr Cu Cucumber	47 Os Cj Cucumber.js	48 Fr Qu Qunit		49 Fr Cp Capistrano	50 Fr Ju JuJu	51 Os Rd Rundeck	52 Os Cf CFEngine	53 Fr Pk Packer	54 Fm Bx Bluemix	
55 En Db DB2	56 Fm Bb Bitbucket	57 Fm Qb QuickBuild	58 En Ub UrbanCode Build	59 Pd Ta Visual Build	60 Fm Tc TeamCity	61 Fm Sh Shippable	62 Os Cc CruiseControl	63 Os Ay Artifactory	64 Fr Ju JUnit	65 Fr Jm JMeter	66 Fr Tn TestNG		67 En Rd RapidDeploy	68 Fm Cy CodeDeploy	69 En Oc Octopus Deploy	70 No CA Nolio	71 En Eb ElasticBox	72 En Ad Apprenda	
73 Fr Cs Cassandra	74 En Hx Helix	75 Os Msb MSBuild	76 Os Rk Rake	77 Os Lb LuntBuild	78 Os Cu Continuum	79 Fm Ca Continua CI	80 Os Gu Gump	81 Os Ng NuGet	82 Os Ap Appium	83 En Xltv XL TestView	84 En Tc TestComplete		85 Os Go Go	86 En Ef ElectricFlow	87 En Xld XL Deploy	88 En Ud UrbanCode Deploy	89 Os Mo Mesos	90 Os Cf Cloud	

Share

Embed

Become Excellent!

Subscribe here!

91 En Xlr XL Release	92 En Ur UrbanCode Release	93 En Ls CA Service Virtualization	94 En Bm BMC Release Process	95 En Hp HP Codar	96 Pd Ex Excel	97 En Pl Plutora Release	98 En Sr Serena Release		99 Fm Tr Trello	100 Pd Jr Jira	101 Fm Rf HipChat	102 Fm Sl Slack	103 Fm Fd Flowdock	104 Pd Pv Pivotal Tracker	105 En Sn Servicenow				
106 En Sp Splunk	107 Os Ki Kibana	108 Fm Nr New Relic	109 Os Ni Nagios	110 Os Gg Ganglia	111 Os Ct Cacti	112 Os Gr Graphite	113 Os Ic Icinga		114 Fm Sl Sumo Logic	115 Os Ls Logstash	116 Fm Lg Loggly	117 Os Gr Graylog	118 Os Sn Snort	119 Os Tr Tripwire	120 En Cy CyberArk				

What is DevOps?

1. DevOps **vague definitions**
2. in the **past** the common practice was **R&D and OPS**, over time,
   the need for something in the middle was born (hence DevOps)
1. Market Definition covers mostly the areas **continuous integration and continuous deployment** and tools like Chef, puppet, ansible, Jenkins etc.
2. My Definition: everything that happens from Developers **commit event until the production** deployment.
3. **regardless of the definition , the essence is clear: create automation, to make the development cycle error free. in the long run will save you , money, and make your development cycles shorter!**

Source control

- GitHub,
- GitLab,
- BitBucket,
- SubVersion
- GIT
- Mercurial
- ClearCase
- TFS

What is DevOps food chain?

1. commit event
2. Continuous Build
3. Continuous Packing
4. Continuous deployment
5. Continuous Testing.

Why continuous build?

1. to fix the “broken build” problem.
2. To avoid the “ אצלי זה עובד“ problem
3. maintain “only” one error free way to compile the code.
4. (gated check-in)
5. E.g team city (java build management and CI)

Build dependencies

A Virtual Environment is a tool to keep the dependencies required by different projects in separate places, by creating virtual Python environments for them. It solves the “Project X depends on version 1.x but, Project Y needs 4.x” dilemma, and keeps your global site-packages directory clean and manageable.

Prerequisites

Ant

Maven

MSBUILD

Why continuous Packing / Integration?

1. Code Version control
    a. Easy to manage.
    b. easy to go back in time to
2. sometimes a package consists of many moving parts, and it is easy to forget/make a mistake
3. Binary version controls such as Artifactory [enterprise grade]

Jenkins

Continuous integration tools

1. Travis : designed for Rails apps, but now supports more than a **dozen languages i**ncluding PHP, JavaScript, C/C++, Objective-C, Python, Haskel, etc. Travis is **free** for all public repositories hosted on **GitHub**. There is **not much to do to configure** - just construct a YAML file to tell Travis what to do with your code.

2. Jfrog artifactory: advanced enterprise grade binary repository with advanced features such as security, caching, proxy etc.

3. Jenkins : open source CI tools , very common, strong community, many plugins. Easy install, easy upgrade, easy to script with (cmd, REST for pythong, xml, json), amazon plugin, maven plugin. Etc.

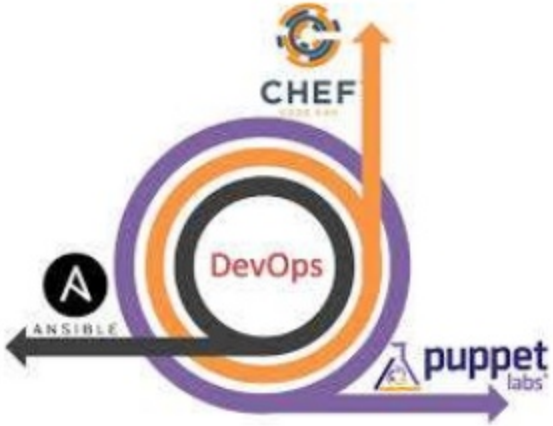
4. **Hudson**.is a continuous integration (CI) tool written in Java

5. CircleCI

6. CodeShip

Why continuous Deployment?

1. to avoid endless **manual steps to deploy** complex clusters
2. to manage the complexity of **configuration management** per **environment** (sometime the biggest problem in the DevOps chain)
3. Manual deployment = **non stable production** system.
4. Comparison



Code Review Tools

SonarQube,

CodeClimate,

Takipiki,

Bliss

What kind of environments?

Google Cloud Platform

SOFTLAYER®
an IBM Company

1. **Dev** (usually developer computer, or central server for team)
2. **QA** - clean environment for development.
3. **Staging** - clean environment as close as possible to the customer setup , 1000 nodes of hadoop etc)
4. **production** (cloud or on site)

amazon web services™

vmware

A word about conternization

- dockers
- rkt,
- LXD.
- There is even a standardization initiative to make sure these containers from different vendors play well with each other.

Orchestration

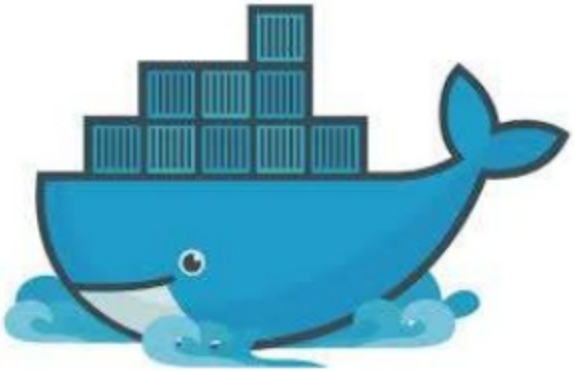
- Kubernetes,
- CoreOS,
- Apache Mesos,
- DC/OS.

Logs

- Logstash
- Fluentd
- Nagios

Where is the QA (continuous testing) ?

1. In some companies, continuous QA (AKA **automation QA**) will be part of the DevOps Process, simply b/c it makes sense. (e.g the continuous testing is so heavy , it can not be manually managed, and the testing is fired as a response to an event.).
2. Depending on the QA process, Complexity , as long as each phase of the Testing is kept, u can change the order as u wish:
   a. **sanity**
   b. **regression**
   c. **stress**
   d. **Performance**
   e. **stability**



when is the QA (continuous testing) ?

1. Hourly?
2. nightly?
3. daily ?
4. weekly?
5. Monthly?
6. all of the above?

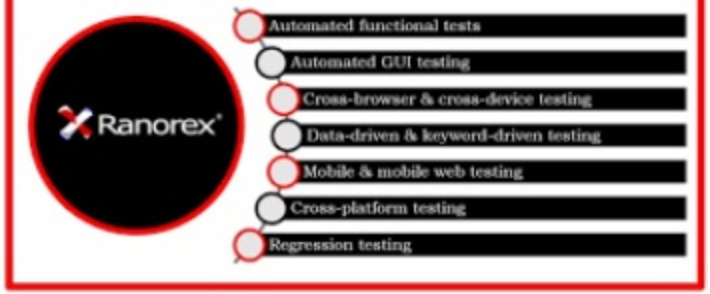
Name:	XYZ Systems
Project ID:	W10978
Application ID/Name:	XYZ Accounting
From Report Date:	11-Apr-15
Report Date:	17-Apr-15
Complete By (Milestone):	08-May-15

Manager:	Ram Ray
QA manager:	Shyam Das

SUMMARY	
Total Test Cases	3
Executed	2
Pass	1
Fail	1
Not executed	1

FUNCTIONAL TESTING							
Test Case ID	Description	Pass/Fail/Not Executed	Test Date	Responsible Developer	Responsible Tester	Comment	Additional Comment (other than QA team)
01	Valid Login	Pass	13-Apr-2015	Developer 1	Tester 1	Login successful	
02	Login Error on invalid Login	Fail	13-Apr-2015	Developer 1	Tester 2	Incorrect error message on failure	
03	Forget Password	Not Executed	13-Apr-2015	Developer 3	Tester 2	NA	

Continuous Testing tools



1. Ranorex: testing framework for desktop,web,mobile applicaiton. Based on .net C# , VB .net. s supports Dekstops forms: Windows, SAP ERP, Qt, WPF UI, .NET and Java, Web pages recorder.

2. Selenium: web browser testing framework.

Big data testing...

1. Non trivial

2. Takes time to run. (big data = long test time)

3. VERY COSTLY

Monitoring

1. Nagios
2. Zabbix
3. DataDog
4. Splunk
5. newRelic
6. Ganglia
7. Cacthi

What should we take into account in terms of mindset?

1. **OS** :
   a. many OS ,the correct one should deployed from the developer level all the way to the production.
   reasons to choose and OS system:
      i. support
      ii. security policy of customer
      iii. enterprise / SMB
      iv. type of hardware: server / consumer
   a. fix a version - should NOT change often.

1. **Prerequisites**
   a. if u can not automate - gather a list of the 3rd party packages you install in one document (the actual yum/apt-get command)
   b. preferably - give one person the responsibility to maintain a script
2. **source code** : make sure you all **compile the code the same way**,
   and decide if the package should include source code
1. **installer**
2. **testing**: **User perspective or system perspective?**

Project and product management Guidelines:

1. **Product** must take into consideration , priority to be allocated as part of the **sprint planning**
2. **Project** must take into account a blocked time periods as part of the sprint last phase (**quality assurance phase**)
3. Successful DevOps strategy implies , the **DevOps team should create infrastructure before the product development**, so the R&D cycle will remain as short as possible and human error free.
4. common practice for **sprint planning** allocate time according to complexity to the below
    a. planning
    b. development
    c. Integration & Quality assurance
5. the **% of time allocated** to each of the below,
   falls under common sense, and may change
6. Jira, Asana, Taiga, Trello,Basecamp, Pivotal Tracker,
   TRAC, MANTIS, redmine,StackStorm

now what, what are the guidelines?

1. Relax and **consult** your **automation/DevOps architect.**
2. each sprint , as part of the retrospective process, take the most **painful manual operations/ error prone process and automate it.**
3. if not possible to automate today, the bare minimum, is to keep the **list of all the linux commands** documented. even in txt document.
4. Try to **anticipate the pitfalls** before you fall into them… (product/project/PO / Scrum master)
5. The **difference** between **good devops and bad devops**
    a. the amount of **maintenance**!!!
    b. after a while - the automation become a **black box**, the only thing u know is what come is (source code) what come out (**PASS**/**FAIL**)

Bottom line:

1. How complex is your **server** management?
2. How complex is your **build**?
3. How complex is your **testing**?
4. How complex is you **deployment**?
5. **Scale**?
6. Existing **Skill set** in your organization and in the market?
7. **Maintenance** overhead?