CS 4320 / 7320
SOFTWARE ENGINEERING

Communication & Configuration Management

WHAT IS THIS CONFIGURATION TO BE MANAGED?

- A **system** is a combination of ***interacting elements*** organized to achieve one or more stated purposes.
- The **configuration of a system** is
- … the functional and physical ***characteristics*** of hardware or software as set forth in technical documentation or achieved in a product.
- … a collection of ***specific versions*** of hardware, firmware, or software items combined according to ***specific build procedures*** to serve a particular purpose.

CONFIGURATION MANAGEMENT

1. Hardware and Firmware Configuration Management

   (but not covering that here…)

2. Software Configuration Management

   Related to Maintenance and Quality Assurance (Testing)

CONFIGURATION MANAGEMENT: WHY?

- It is about ***change***
- **Considering a change – helps you consider and answer:**
- What dependencies are there?
- Does the proposed change break anything?
- Will the system still fulfill its stated requirements if we make the change?
- **After a change, if you didn't do the above well:**
- Ooops. Something broke - what have we changed and when?

CONFIGURATION MANAGEMENT: WHAT?

- Is a discipline applying technical and administrative direction and surveillance to:
  1. **Identify and document** the functional and physical characteristics of configuration items
  2. **Control changes** to those characteristics
  3. **Record and report change** processing and implementation status
  4. **Verify compliance** with ***specified requirements***

CONFIGURATION MANAGEMENT: COMPARING WHYS AND WHATS

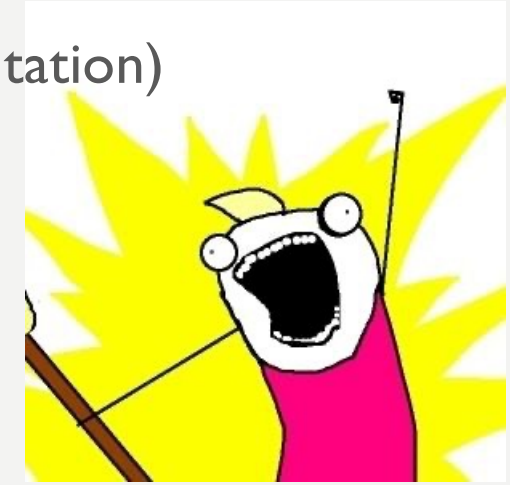
- **Before a change:**
  1. What dependencies are there?
  2. Does the proposed change break anything?
  3. Will the system still fulfill its stated requirements if we make the change?
- **After a change gone wrong:**
  1. Something broke - what have we changed and when?

1. **Identify and document** the functional and physical characteristics of configuration items
2. **Control changes** to those characteristics
3. **Record and report change** processing and implementation status
4. **Verify compliance** with ***specified requirements***

IDENTIFY AND DOCUMENT

True Story: Server OS version X will no longer be supported. We need to upgrade and move all our applications from those servers. Do we know what applications are living on our servers with version X? No?

1. Need to know all the things (electronic documentation)



What things? Requirements, design, tests,
source and executable code, version definitions,
environments, tools, licenses,
installation and operations docs, …

IDENTIFY AND DOCUMENT

2. Documentation must be up-to-date

   Processes must exist and be followed/enforced to update documentation

2. Documentation should be accessible, usable, searchable

   Documentation that isn't used is a waste

IDENTIFY AND DOCUMENT

- Relationships are particularly important

1. Requirements <–> Design <–> Code traceability, with versioning
   How? Identification codes, configuration tools that link
2. Structural dependencies
   Design documentation, deployment documentation

CONTROL CHANGES

True Story: There is a bug in application Y. Where is the source code? Kim coded and deployed it before she went to another company…but never checked it in, and her PC has been wiped and reassigned. So version control does not have the production copy…

True Story: The application has been maintained without a view to the overall architecture. It now resembles the proverbial big ball of mud. A change in one place breaks it in 10 others – people are afraid to touch it.

True Story (more or less): Jim pushed the new release into production Friday afternoon the day before a big marketing push, and the application broke under the load. And Jim left for vacation Friday night.

CONTROL CHANGES

1. Processes for requesting, evaluating and approving changes
2. Processes for designing, coding, testing, releasing, and deploying changes

RECORD AND REPORT CHANGES

- Processes
- Tools
  - Version Control tools
  - Build tools with history
  - Change control tools

Verify Compliance

- Management oversight
- Audits

CURRENT IDEAS IN SWE: CONTINUOUS DELIVERY AND DEVOPS

- Both ideas focus on the rapid, frequent, and reliable delivery of working software.

  What else does that sound like?

- They are not the same, but they support each other
- They REQUIRE good Configuration Management

CONTINUOUS DELIVERY

- A Software Engineering approach in which
  - teams produce software in short cycles, building, testing, and releasing software faster and more frequently.
  - the software can be reliably released at any time
  - cost, time, and risk of delivering changes is reduced through **incremental updates** to applications in production.
- **A straightforward and repeatable deployment process is important for continuous delivery.**

DEVOPS

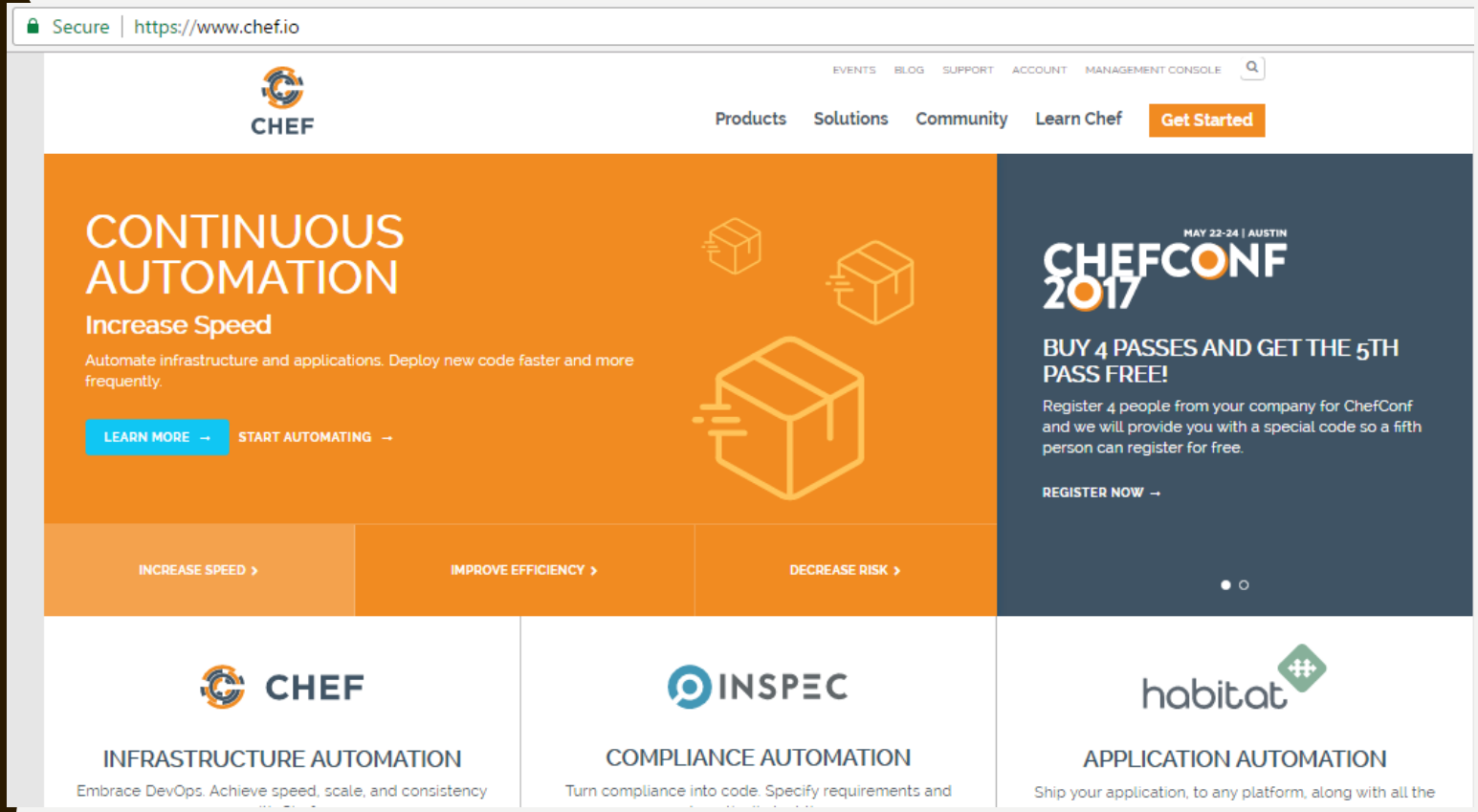
- A set of practices that…
- ➢ Emphasize the collaboration and communication of both software **developers and IT** professionals
- ➢ **Automate the process** of software delivery and infrastructure changes
- ➢ Aim at establishing a **culture and environment** where building, testing, and releasing software can happen rapidly, frequently, and more reliably

POPULAR TOOLS FOR CM / CD

https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software

	Language	License	Mutual auth	Encrypts	Verify mode	Agent-less	Have a GUI	First release	Latest stable release
Ansible	Python	GPLv3+	Yes[1]	Yes[2]	Yes	Yes	Yes[3] (Free 30-day Trial)	2012-03-08	2016-11-01 2.2.0.0[4][5]
Bcfg2	Python	BSD 2-clause[6]	Yes[7]	Yes[8]	Yes[9]	No	Yes[10]	2004-08-11[11]	2014-09-05 1.3.5[11]
Capistrano	Ruby	MIT License		Yes[2]				2005	2015-03-02 3.4.0
cdist	Python	GPLv3+	Yes[1]	Yes[2]		Yes		2010	2015-03-19 3.1.12
Chef	Ruby, Erlang	Apache 2.0	Yes[12]	Yes[13]	Yes[14][15]	No	Yes	2009-01-15 0.5.0	2016-03-07 12.8.1 (client),[16] 2016-02-04 12.4.1 (server)[17]
CFEngine	C	GPLv3[18]	Yes[1]	Yes[19]	Yes[20][21]	No		1993	2016-10-31 3.10.0b1
ISconf	Python	GPL[22]	Yes[23]	No[24]				1998	2006-08-13 4.2.8.233
Juju	Python, Go[25]	Affero General Public License	Yes[1]	Yes[8]	No	No	Yes[26]	2010-09-17[27]	2015-06-17 1.24.0[28]
Local ConFiGuration system (LCFG)	Perl	GPL	Partial[29]	Partial[30]				1994	Weekly Releases
NOC	Python	BSD	Yes[1]	Yes[2]	Yes	Yes	Yes	2012-03-08	2015-05-20 15.05.1[31]
OCS Inventory NG with GLPI	Perl, PHP, C++	GPL	No[32]	Yes[8]		No		2003	2014-07-13[33]
Open pc server integration (Opsi)	Python, Java	GPL	No	Yes[8]		No		2004	2013-03-01 4.0.3
PIKT	C	GPLv2+[34]	Yes[35]	Yes[36]		No		1998[37]	2007-09-10 1.19.0
Puppet	Ruby	Apache from 2.7.0, GPL before then	Yes[38]	Yes[8]	Yes[39][40]	No	Yes[41]	2005-08-30[42]	2016-03-16 4.4.0[43]
Quattor	Perl, Python	Apache 2.0[44][45]	Yes[46]	Yes[47]				2005-04-01[48]	2017-03-03 17.2.0[49]
Radmind	C	BSD[50]	Yes[51]	Yes[52]		No		2002-03-26[53]	2008-10-08 1.13.0[54]
Rex	Perl	Apache	Yes[1]	Yes[2]		Yes		2010-11-05 0.9.0[55]	2015-09-04 1.3.3[56]
Rudder	C and Scala	GPLv3 and Aapche 2.0[57]	Yes[1]	Yes[8]	Yes[58][59]	No	Yes	2011-10-31	2016-12-20 4.0.2[60]
SmartFrog	Java	LGPL	Yes[61]	Yes[61]		No		2004-02-11	2009-01-26 3.16.004[62][63]
Salt[64]	Python[65]	Apache 2.0[66]	Yes[67]	Yes[67]	Yes	Both[68][69]	Yes[70][71]	2011-03-17 0.6.0[72]	2016-06-08 2016.3.1[73]
Spacewalk	Java (C, Perl, Python, PL/SQL)	GPLv2	Yes	Yes		No		2008-06[74]	2016-11-29 2.6[75]
STAF	C++	CPL[76]	No[77][78]	Partial[79]		No		1998-02-16[80]	2012-12-16 3.4.16 [81]
Synctool[82]	Python[83]	GPLv2[84]	Yes[85]	Yes[2]	Yes[86]	Yes[87]		2003[88]	2014-06-15 6.1[89]
	Language	License	Mutual auth	Encrypts	Verify mode	Agent-less	Have a GUI	First release	Latest stable release

POPULAR TOOLS FOR CM / CD : CHEF



POPULAR TOOLS FOR CM / CD : PUPPET

Secure | https://puppet.com/product

puppet

Product Solutions Services Resources Community Partners Company

Puppet Enterprise 2017.1 is here Now it's easier than ever to scale Puppet across teams and environments **See what's new →**

Drive change with confidence with Puppet Enterprise

Puppet Enterprise is the leading platform for automatically delivering, operating and securing your infrastructure – no matter where it runs. With Puppet you know exactly what is going on with all your software. And you get the automation needed to drive change with confidence.

The result: you can deliver more software faster than ever, while maintaining quality, security, and compliance.

See how Puppet works

TOO SMALL FOR THAT? ROLL YOUR OWN

1. Establish your own processes
2. Use version control
3. Automate as much as possible (deployment, etc.)
4. Document with wikis, spreadsheets, linked documents
5. Remember the Whys