Building Your Continuous Delivery Toolkit

T.j. Randall
XebiaLabs
xebialabs.com
@xebialabs

June 18, 2014

#jenkinsconf

Why a discussion about "other tools" at a JUC conference?

- It's ok to think you may need additional tooling to solve your issue for everything that you need to do.
- Clearly the reason you expanded Jenkins in this area is that it's a great tool, and you can do anything with Jenkins!
- You're hitting some challenges:
  - Maybe you've built a solution, it's running in PROD, and you're having challenges maintaining it.
  - Or, maybe you want more, and you can't take out what you've currently built because it solves some specific needs.

What do we use?

We love Jenkins ourselves and use it for our continuous integration needs

- **Build Failure Analyzer**
  - Helps us to understand why builds are failing, especially when analying possible infrastructure issues.
- **Matrix Reloaded Plugin**
  - Allows users to easily rebuild parts of an already built Matrix build.
- **Promoted Builds Plugin**
  - Distinguish good builds from bad builds by introducing the notion of 'promotion'.
- **Throttle Concurrent Builds Plugin**
  - For throttling the number of concurrent builds of a project running per node or globally.

What do we see others use?

I get the benefit to go to lots of different customers, in all kinds of verticals, and see how they are using Jenkins.

- **Build Monitor Plugin**
  - Great for teams to publish build results on screens
- **Swarm Plugin**
  - Enables slaves to auto-discover nearby Jenkins master and join it automatically, thereby forming an ad-hoc cluster.
- **Shelve Project Plugin**
  - Lets you shelve projects so that they can easily be resurrected. (delayed trashcan ;)
- **Timestamper**
  - Adds timestamps to builds. Makes it easier to understand what is running slowing, or see that a job is stuck without activity.

When do some people start to think about other tools

When do you hit the edges of your CI tool?

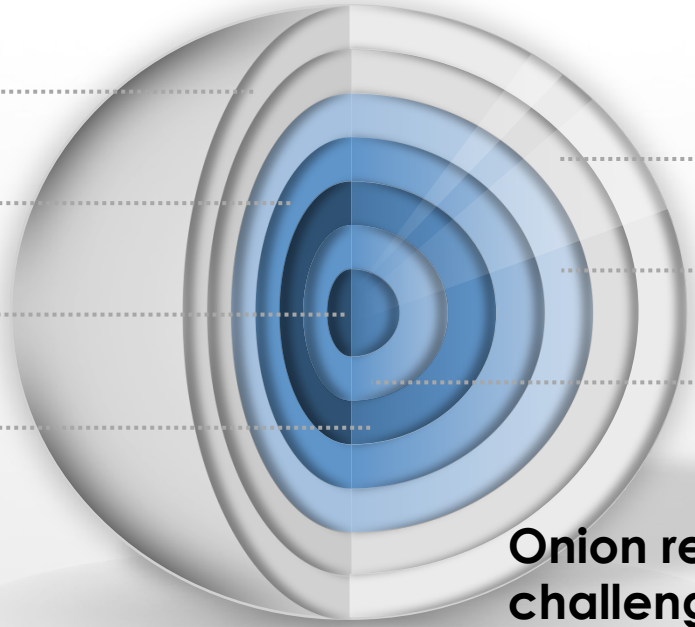
- Security: authentication in DEV, QA & PROD from CI tool
- Reporting: providing necessary documentation for release

Jenkins has the luxury problem based on its success

- People have expanded technically, and are reaching limits (“everything is possible”)
- Expanding into areas where they hit organizational problems

The Onion of Continuous Delivery

The onion is bigger than Jenkins - in each layer is more "pain" (but not necessarily in this order)

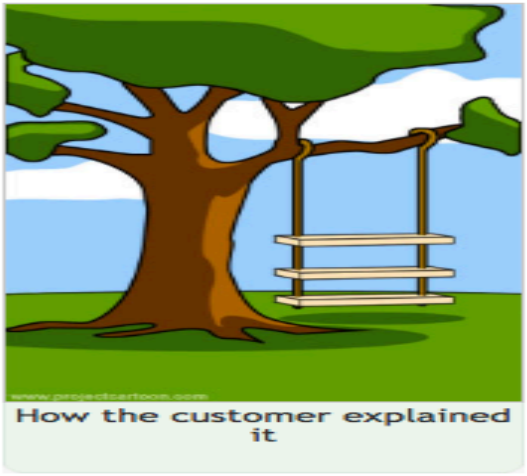


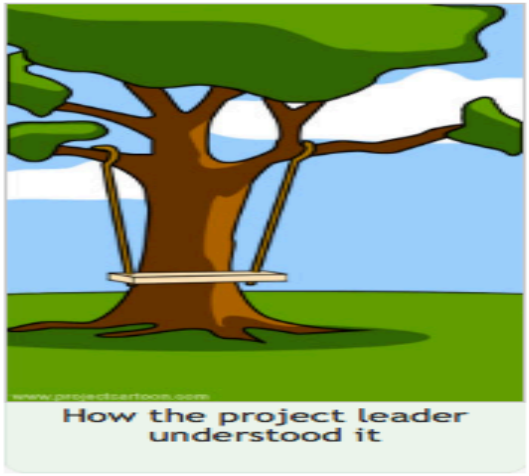
Onion represents technical challenges, as well as multiple parts of the organization

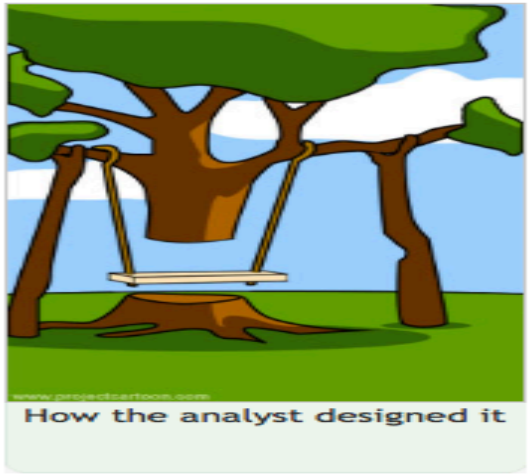
- Consumer audience not happy with how information is presented

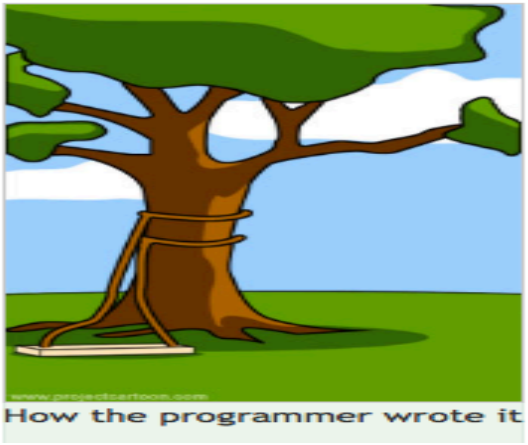
What do you need for Continuous Delivery?
(as it relates to teams)

Ever notice that our CD tooling is a lot like this?









How do you "Productionalize" a Toolkit

- How do you include all of your teams in your continuous delivery tool set? How did your automation tools get chosen?
- Who maintains your automation tools?
  - Upgrades? Compliance?
- Most organizations try to **limit/avoid** cross-team activities when choosing a tool.

Outline for Continuous Delivery Tools

Let's take a look at five technical needs for continuous delivery as they relate to both tools and teams.

- Continuous Integration
- Deployment
- Scaling
- Test
- Release Management

For the conversation, we'll use five teams that you might come across as part of your application release process.

- DEV
- QA
- DBA
- Network Services
- Ops

Continuous Integration (reality)

Team	What they currently have
DEV	CI tool that allows them to quickly build applications, as well as many other DEV teams normally drive this decision
QA	*"CI tool? We don't need no stinkin' CI tool!"*
DBA	*"I don't check SQL into a SCM."*
Network Services	*"What's a CI tool?"*
OPS	*"What's a CI tool?"*
Business	*"What's a CI tool?"*

VERY hard to answer "What Changed?", "Why Changed?" and "How Changed?"

Continuous Integration

Team	What they need
DEV	Same CI tool for all teams. Commits can document **why** a change was made. *(not just "fixed variable assignment")*
QA	QA needs to understand the changes as the relate to request (enhancement, bug fix, etc.)
DBA	SQL scripts are code. Check them in.
Network Services	Scripts are code. Check them in.
OPS	Can use CI information for understanding parts of system that are changing.
Business	Visibility into the delivery of their changes

Deployment (reality)

Team	What they currently have
DEV	Some tools, some custom scripting, some magic sauce
QA	*"You're not putting that tool in **MY** environment!"*
DBA	*"You can't automated database deployments. Just call us when you're ready."*
Network Services	*"There's a deployment going on? Why didn't anyone tell us?!"*
OPS	*"Why are there seven tabs of instructions for one app?!"*
Business	*"Can someone please tell us when we can log in to test?"*

Different deployment tools and processes across environments

Deployment

Team	What they need
DEV	Tooling that ensures consistent, fast deployments in environments.
QA	Deployments based on same process as DEV, with no manual steps.
DBA	Either SQL code is part of automated deployment, or notifications included in overall automation.
Network Services	Visibility into current and upcoming deployments
OPS	Same deployment process as earlier environments, with no manual steps
Business	Visibility into current and upcoming deployments, along with automated notifications.

Scaling (some versions of reality)

Team	What they currently have
DEV	Scripting to instantiate needed environments
QA	Requests made to Network Services for infrastructure
DBA	*"You can't scale databases!"*
Network Services	Service entity for creating, maintaining and granting access to infrastructure
OPS	*"We've got our own environments to maintain"*
Business	*"What address do I put in the browser?"*

Difficulty creating, maintaining and accessing scaled infrastructure

Scaling

Team	What they need
DEV	Infrastructure scaled as part of CI / deployment process
QA	Infrastructure scaled as part of deployment process
DBA	Hop in the pool! The water is great!
Network Services	Visibility / access to all infrastructure
OPS	Visibility / access to all infrastructure
Business	Clear information about infrastructure available for access

Test (reality)

Team	What they currently have
DEV	CI tools provide a lot of functionality here. Tests not related to other groups (to help avoid duplication)
QA	Normally choose/maintain their own test tools. Tests are created based on conversations with DEV and business. QA is the first "integrated" environment.
DBA	*"Testing? The database is up…."*
Network Services	Normally choose/maintain their own tools, which are not accessible/used by other teams
OPS	Ad hoc reports of tests completed
Business	*"We'll do our own "smoke" (manual) testing, and track the results in a spreadsheet"*

Silos of information, as well as non-consistent testing across environments

Test

Team	What they need
DEV	Test tool that gives quick results on each build. Same tests are available to other teams for their use
QA	Tooling that gives them much broader coverage/feedback, as well as testing in earlier environments.
DBA	Database needs to be tested also – **not** just load testing!
Network Services	Testing application as it relates to infrastructure in each environment
OPS	Tests that validate release is successful
Business	Feedback into all groups about testing requirements, so that more automated tests can be built

Release Management (aka "Excel")

Team	What they currently have
DEV	Some tooling (ex/ JIRA); E-mails sent to PM to relate what is being worked on
QA	Sources of information (e-mail, defect system, spreadsheets, etc.) to relate what is in QA / what is coming.
DBA	*"Just tell us when to deploy your SQL."*
Network Services	*"I've got my own releases to maintain – I can squeeze you in next Thursday."*
OPS	*"Wednesday release meeting (kinda) tells me what to expect this week-end."*
Business	*"When are my features getting delivered?!"*

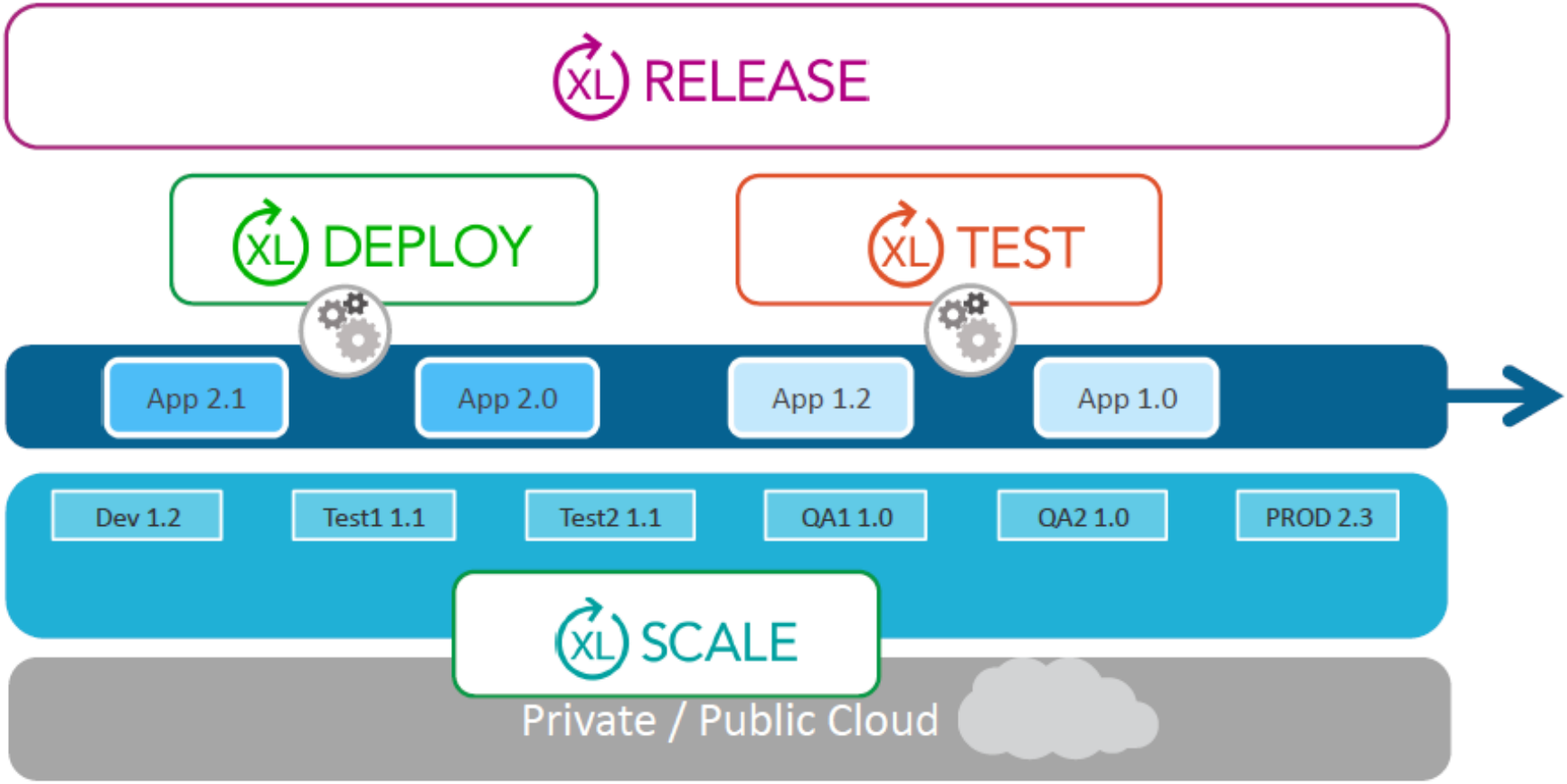
Kludge, manual, broken communication for the most important thing!

Release Management

Team	What they need
DEV	CI, deployment, scaling and testing information feeds into release management system
QA	Release management drives dates so that QA is part of overall release, not viewed as a bottleneck
DBA	Team actively participates in their portion(s) of the release process
Network Services	Included, but not necessarily actively involved, with **application** release
OPS	Easily access all information about **all** releases.
Business	Ability to **access** and **participate** in release process

(not so) Clever closing…

Did I mention I was a vendor?



Thank You To Our Sponsors

Platinum

CloudBees

Gold

black diamond SOFTWARE

MidVision™ Release the innovation

choose digital

JFrog

XebiaLabs the deployment automation company

Silver

SOASTA Test Faster. Release Sooner.

ELIASSEN GROUP Agile Consulting Services

KENSHO

SERENA™

assembla