From 248 to 98 then 1
Our pivot to continuous delivery

Jean-Michel Lemieux
Atlassian
jml@atlassian.com
@jmwind
Why now?
"Continuous delivery is not a cowboy show. It puts you in charge of your production environment.

The business can pick and choose what and when to deploy.

If you think you’ve nailed agile development, but aren’t considering how to achieve continuous delivery, you really haven’t even started."

- ThoughtWorks Tech Radar.
248 to 98
248 to 98

- **Goal**: shorten our release cycles
248 to 98

- **Goal**: shorten our release cycles
- **Success**: better efficiency, better releases, roughly 90 days.
248 to 98

- **Goal**: shorten our release cycles
- **Success**: better efficiency, better releases, roughly 90 days.
- **Failure**: crap releases, shipped on time no features, or no shortening of time. api/integration madness.
## Baseline

<table>
<thead>
<tr>
<th>Tool</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>JIRA</td>
<td>248</td>
</tr>
<tr>
<td>Bitbucket</td>
<td>30</td>
</tr>
<tr>
<td>GreenHopper</td>
<td>34</td>
</tr>
<tr>
<td>Confluence</td>
<td>185</td>
</tr>
<tr>
<td>Studio</td>
<td>72</td>
</tr>
<tr>
<td>FE/CRU</td>
<td>192</td>
</tr>
<tr>
<td>Bamboo</td>
<td>142</td>
</tr>
<tr>
<td>Crowd</td>
<td>149</td>
</tr>
</tbody>
</table>
Recap
Training Plan v2
98 to 1
98 to 1

- **Goal**: allow products to independently release to hosted (eg, production) every day
98 to 1

• **Goal**: allow products to independently release to hosted (eg, production) every day

• **Success**: better efficiency, faster learning, happier/more customers.
98 to 1

• **Goal**: allow products to independently release to hosted (eg, production) every day

• **Success**: better efficiency, faster learning, happier/more customers.

• **Failure**: crap releases, api/integration madness, slowing down because of too many balls in the air.
Approach
Approach

Look at the entire system
Approach

Look at the entire system
If it’s hard, do it more often
Approach

Look at the entire system
If it’s hard, do it more often
Measure and remove the waste everywhere
Approach

Look at the entire system
If it’s hard, do it more often
Measure and remove the waste everywhere
Approach

Look at the entire system
If it’s hard, do it more often
Measure and remove the waste everywhere
Approach

Look at the entire system
If it’s hard, do it more often
Measure and remove the waste everywhere
System Breakdown

Product Architecture → Analytics / Monitoring → Customer Feedback

Culture / Process → Pipeline

OnDemand → OnSite
System Breakdown
System Breakdown

- The hosting platform (e.g., production).

Pipeline

OnDemand

OnSite

Analytics / Monitoring

Customer Feedback
System Breakdown

- The hosting platform (eg, production).
- Tools and processes to move products through to production.
System Breakdown

- The hosting platform (eg, production).
- Tools and processes to move products through to production.
- Tools to monitor deployed products.
System Breakdown

- The hosting platform (e.g., production).
- Tools and processes to move products through to production.
- Tools to monitor deployed products.
- Tools to stage, test, and ensure we don’t fuck the customer.
System Breakdown

Culture/Process

OnDemand

OnSite

Analytics / Monitoring

Customer Feedback
System Breakdown

Culture/Process

- Development principles

OnDemand

OnSite

Analytics / Monitoring

Customer Feedback
System Breakdown

Culture/Process

- Development principles
- Team structures

OnDemand

Analytics / Monitoring

Customer Feedback

OnSite

Thursday, 5 April 12
System Breakdown

Culture/Process

- Development principles
- Team structures
- Planning process

OnDemand

Analytics / Monitoring

Customer Feedback

OnSite
System Breakdown

Culture/Process

- Development principles
- Team structures
- Planning process
- Development process

OnDemand

Analytics / Monitoring

Customer Feedback

OnSite
System Breakdown

Product Architecture

Analytics / Monitoring

Customer Feedback

OnDemand

OnSite
System Breakdown

- Technical coupling between products and addons.

Product Architecture

Analytics / Monitoring

Customer Feedback

OnDemand

OnSite
System Breakdown

- Technical coupling between products and addons.
- UI Integration patterns/goals
System Breakdown

Product Architecture

- Technical coupling between products and addons.
- UI Integration patterns/goals
- Shared services, APIs, and libraries

OnDemand

OnSite

Analytics / Monitoring

Customer Feedback

Thursday, 5 April 12
Geek details

- Split compute from storage. No disks in compute. RAM. Less power. 1s startup.
- Data centres only provides power and floor space. We own everything else.
- Ditch full virtualization used openVZ.
- Partnered with an integrator.
- Shared mount to avoid rsync. 500-700 customers per rack. Deploy one set of binaries.
- Allows regional rollouts, daily upgrades, fast restarts
- and much more.... dark features, plugins, manager nodes, backups...
Monitoring

Prioritizing rules:
- Increase Uptime: (detect better) MTTD -> enhances SLA
  (resolve faster) MTTR -> enhances SLA
- Prevent/Predict Failure
  MTBF

COLLECTION (metrics/events)
- Gmond
- Collectd
- Nagios
- Analytics
- MetaOps (deplays, syncs, etc.)
- New Relic
- Logsstash
- Statsd
- Google Analytics
- HIT Metrics

TRANSPORT
- RabbitMQ
- MSB
- 0MQ
- P2P Tracker

CHECKS/ALERT (is memory < 90%)
- Nagios Check
- Opentsdb check on steroids
- CEP (Esper)
- Opentsdb on steroids
- Http check on steroids

NOTIFICATION LOGIC
- some logic

NOTIFICATION
- Zendesk
- Hopsbot
- Nagios Alert
- Email
- MetaOps
- MSB

MONITORING OVERVIEW 19/01/2012

DEPLOY
- Dashboard
- Triage Tool
- Graph Dashboard

STORAGE
- Event Storage
- Opentsdb
- Graphite

EXTERNALIZED API
- Graph API
- Status API
- Extract Data
- Ingest Data
- Notification API
- Search Data
- Event API

LOGSTASH
- Elastic search
- Meta Facts
Monitoring Collection

Visualization

Storage

Filtering

Notification

REST API

Prioritizing rules:
- Increase Uptime:
  - MTTD -> enhances SLA
  - MTTR -> enhances SLA
- Prevent/Predict Failure
  - MTBF

Thursday, 5 April 12
Process and Culture
Process Changes

- Small shipable chunks. Decoupled from one another (includes marketing)
- Minimize hand-offs
  
  Product teams deploy themselves.
  
  X-team feature teams PM/Dev/Marketing
- Minimize branching, use dark features and plugins
- Upfront testing, done means done
- Metrics and goals for theme teams allows more asynchronous, measurable, cradle to grave.
Product Architecture
Architecture Principles
Architecture Principles

Products
Architecture Principles

Products

- Stay away from a mega-app
- Assume products are not hosted on the same physical infrastructure
- Use integration levels and make them opt-in and backwards compatible.
Architecture Principles

Addons

Stay away from a mega-app
Assume products are not hosted on the same physical infrastructure
Use integration levels and make them opt-in and backwards compatible.

Products
Architecture Principles

Addons

- Independant Deploy
- Rich UI Integration
- Independant Purchase
- Own User Licensing
- Remote
- Can’t slow products

Products

- Stay away from a mega-app
- Assume products are not hosted on the same physical infrastructure
- Use integration levels and make them opt-in and backwards compatible.
Cross Training
Cross Training
Cross Training

MVP 1.0

Payed
Beta
Cross Training

Discounts

MVP 1.0

Payed

Beta
Cross Training

MVP 1.0

Discounts

Fast Releases

Payed Beta
Cross Training

- MVP 1.0
- Payed Beta
- Discounts
- Fast Releases
- Wrong Market?
Cross Training

- MVP 1.0
- Payed Beta
- Discounts
- Fast Releases
- Wrong Market?
- Killer Features
248 to 98 to 1
248 to 98 to 1-7
Recap

• How do you compare?
  • Someone is running in front of you
• Holistic systems thinking
  • Needs a company wide pivot, decision making has to change
  • Build your Unicorns (think 10x)
• Architecture is important to scale
  • Share your principles and system view so that teams can self optimize and remove waste.
• Cross train
  • Try several small experiments
Thank You!