How to build and use a pipeline factory to accelerate your DevOps journey

Peter Maddison
Who am I?

Peter Maddison

- Design Engineer – High frequency microwave electronics
- SA and beyond – Individual contributor running the DC, DBA; moved to Canada and built a high-performance team working in FI
- Out of banking – Project manager building DR, service desk, ITSM, AD then program manager moving datacenters; finally Director running Platform Engineering
- Consultant – First on my own and now building Xodiac
I get paid for taking large, complex ideas and making them succinct. This presentation is the opposite of that...

- Peter Maddison
Agenda

• What problem are we solving?
• What does the solution look like?
• How do you create a model that works for you?
• How does this all work?
• There are so many options... What to use?
• How do you get teams to come and use it?
• What have we learned?
I want to hear from you
http://sli.do/#B894
What problem are we solving?
Process

We *want* this

Development ➔ Production ➔ Feedback

But we really *need* this

Development ➔ QA ➔ Security ➔ Prod ➔ Ops ➔ Feedback
Process

We need this

Development > QA > Security > Integration > Operations

But you could say we really want this

Development > Organizational concerns addressed > Operations
Common concerns

- How do we know our code is appropriately tested?
- How do we ensure we introduce no security vulnerabilities?
- How do we ensure the pipeline only pushes code that satisfies our organizational controls?
Complexity
Introducing change

“This year, I resolve to stay away from unnecessary risks.”
Easy, right?
Not to mention SDLC...

- Did we build what we said we would?
- Can we show that at speed?
- How do we ensure the result can be supported?
Managing complexity

- To go faster, and solve these problems, we build integrated test environments like these...

- *If teams are actively working around the solution, it probably isn’t the right solution.*
What are the biggest hurdles in your environments?
http://sli.do/#B894
What does a solution look like?
"It’s not all sunshine and roses, but a good amount of it actually is"

- Anonymous
Part of a larger whole
Opinionated delivery
Pipeline overview

Channels (forums, focus groups, social media)

Customers

“Business”

Product Owner

Team members

Delivery Team

Automated Pipeline

Build Test Deploy

Feedback (monitoring, logging, test results, chatops)

Time to ideation

Time to deliver

© 2018, Xodiac Inc. All rights reserved.
Start by understanding

• Where are we today?

• Understand all the steps it takes to get code into production and operational.

• What system are we interacting with? Who is involved? What signoffs are required?
Focus on the delivery

1. Define your controls
2. Select your tools
3. Build your pipelines
4. Start with one, iterate
5. Bask in the glory
Automation

• To create consistency we need to automate
• To execute at speed we need to automate
• To improve quality we need to automate
Trust

• Do we trust our automation?

• Initially, probably not. Need to start small and build that trust.

• Story time...
Getting all cloudy up here
How do you create a model that works for you?
Considerations

We know what happened in the pipe

Our delivery process is secure

Validate the payload in the pipe

Record the execution of the pipe and monitor the product

Traceability

Access

Compliance

Operations

1. Define a system of record
   • Audit your pipelines
   • Report on what is happening

2. Secure the running of the pipe
   • Secure the integrations of the pipe

3. Test for quality
   • Test for security
   • Test for non functional reqs

4. Log metrics
   • Update CMDB
   • Update monitoring systems
Creating a pipe

Portal -> Identity
Example
Repository
Project
Repository
Pipeline
Running the pipe

1. Define work here
2. SCM
3. Build Results
4. Quality tests
5. Artifacts
6. CI run 2
7. Organisational tests
8. Service state

Cycle time
Auditing the pipe

Alternatively to registering back to the work item, we can use visualization of log data.
Where are you running into difficulty?

http://sli.do/#B894
How does this all work?
Scalable pipeline creation...

Requester

Portal interface (select pattern, verify credentials, create JSON)

JSON

DevOps pattern API

1) Identify pattern from JSON
2) If valid, create pipe instance in store, each component tagged as pending
3) Create events for each component with dependencies

Configuration state store

Create update

Pull pipeline elements

Dashboard

Visualize the pipeline metrics

E.g. For Convasa we might structure the state store as:

<table>
<thead>
<tr>
<th>Pipe_ahe_1</th>
<th>Github</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipe_bik_1</td>
<td>Jenkins</td>
</tr>
<tr>
<td>Pipe_ahe_1</td>
<td>UCD</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Pipe_dkt_n</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Pipe_vet_n</td>
<td>...</td>
</tr>
</tbody>
</table>

Cloud Services (BlueMix, AWS, Azure)

Infrastructure (e.g. AD, FS, IDI)

DevOps Tools (e.g. Github, Jenkins, UCD)

Create
... and removal
Now my pipeline is an object...

- I can tell which pipes are using which testing systems and replace them.
- I can reduce complexity by understanding the variants in place.
- I can keep my environments cleaner
Blueprints

1 Select
   • Select the type of application
   • Select the language

2 Create
   • Create the necessary artifacts
   • Integrate and create pipeline

3 Deliver
   • Provide back to the team

For example

Delivery team wants a microservice using Javascript

The blueprint creates all the necessary artifacts and integrates them

Once complete, it returns notification to the developer of how to access his new pipeline
There are so many options…
What to use?
Let’s talk vendors

Create

Execute

Store

Deploy and Configure

Targets
We may need to remain agnostic.

- Just think about the cost of switching
- Or about where your data is.
What tools do you use today?
http://sli.do/#B894
How do you get teams to come and use it?
And cons...
When is it worthwhile?
What have we learned?
"We cannot solve our problems with the same thinking we used when we created them."

- Albert Einstein
So let’s review

• Problem of complexity is difficult to overcome
• Looked at a possible solution by making delivery opinionated
• Reviewed a framework for identifying controls
• Discussed how to build a system to create opinionated pipes
• Considered the tools we could use
• Examined what the impact of doing this could be
Questions?

peter.maddison@xodiac.ca
@pgmaddison
https://www.linkedin.com/in/peter-maddison-1800352/
Slide graveyard