Securing your pipes with a TACO

Peter Maddison
Who am I?

Peter Maddison
Coach, consultant, founder...

peter.maddison@xodiac.ca
@pgmaddison
https://www.linkedin.com/in/peter-maddison/
“I cannot teach anybody anything. I can only make them think.”

- Socrates
Agenda

- What problem are we solving?
- What does a solution look like? (Hint: It involves a TACO)
- How do you create a model that works for you?
- What have we learned?
What problem are we solving?
Introducing change

“This year, I resolve to stay away from unnecessary risks.”
Pipeline overview

Channels (forums, focus groups, social media)

Customers

Product Owner

Team members

Delivery Team

Automated Pipeline

SCM

App
Test
Infra

Build
Test
Deploy

Feedback (monitoring, logging, test results, chatops)

Time to ideation

Time to deliver

1. Auto triggered
2. Manually triggered

A
B
C
Many competing concerns

Developers

Security

Operations

Compliance

Testing
What does a solution look like?
Modeling

Identify what happened in the pipe

Secured delivery process

Validate the payload in the pipe

Record execution and monitor

**Traceability**

1. Define a system of record
2. Audit your pipelines
3. Record what is happening

**Access**

2. Secure the running of the pipe
3. Secure the integrations of the pipe
4. Secure who runs the pipe

**Compliance**

3. Test for quality
4. Test for security
5. Test for non-functional reqs

**Operations**

4. Log metrics
5. Update CMDB
6. Update monitoring systems
## Traceability example

### Purpose

To ensure that for a given request for change, we have a valid chain of custody from the change of functionality to the implementation of the change.

### Control

All production deployments must have a ticket number. Developers must put the ticket number into the submitted pull request in order for the request to be pushed through to production. All ticket numbers since last production deploy must be included in the pull request.

<table>
<thead>
<tr>
<th>Item</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artifact</td>
<td>Ticket</td>
</tr>
<tr>
<td>Location</td>
<td>Jira</td>
</tr>
<tr>
<td>Happy path</td>
<td>Pull request contains valid ticket number</td>
</tr>
<tr>
<td>Sad path</td>
<td>If PR doesn't contain ticket number, build proceeds but only deploys to dev.</td>
</tr>
</tbody>
</table>
OH NO!
I FORGOT ...
SOMETHING ...
... BUT WHAT?
What happened

• Putting hooks into place to trace all activity back to Jira
• Ensured versioning of all pipeline components
• Automated updates to master change system
Traceability Acess Cmpliance Operations

Access
Were the right people involved

• Build once, use many times
• Reduce the size of the change
• Automated deployment
Traceability  Access  Compliance  Operations

Compliance
Have we met our obligations

- Standard pipelines and/or components
- Scan early and track addressing issues
- Separate organizational concerns
Traceability  Access  Compliance  Operations
Will it work tomorrow

- Monitoring for all!
- Enablement of reduced testing cycles
- Decreased MTTR
All together now

Traceability
- Chain of custody
- Test results for all
- Deployed version is tracked
- Change is recorded

Access
- Source code managed
- Creator tracked
- Build once, deploy many
- Pipelines only

Compliance
- Peer review
- Scan the code
- Scan the artifact
- Manage the data

Operations
- Validate the target
- Validate quality
- Check it works
- Watch it live
TACO!
How do you create a model that works for you?
Mapping the controls

If controls fail, break the build and radiate back to team for resolution
Running the pipe

1. Define work here
2. SCM
3. Quality tests
4. CI build
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.
Blueprints

1. Select
   - Type of application
   - Language

2. Create
   - 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
Now my pipeline is an object...

- It can be versioned and treated as code
- We can abstract the systems it calls
- I can keep my environments cleaner
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

- A way to create common understanding of DevOps pipelines
- An approach to build more secure software delivery pipelines
- Ways to help ensure software delivery compliance
- A framework to drive good DevSecOps practices
Free 5 minute management quiz on how ready you are for your digital transformation: https://www.xodiac.ca/quiz

Questions?

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