

Continuous Delivery Deployment Pipelines

How to Build Better Software Faster

Dave Farley



<https://www.davefarley.net>



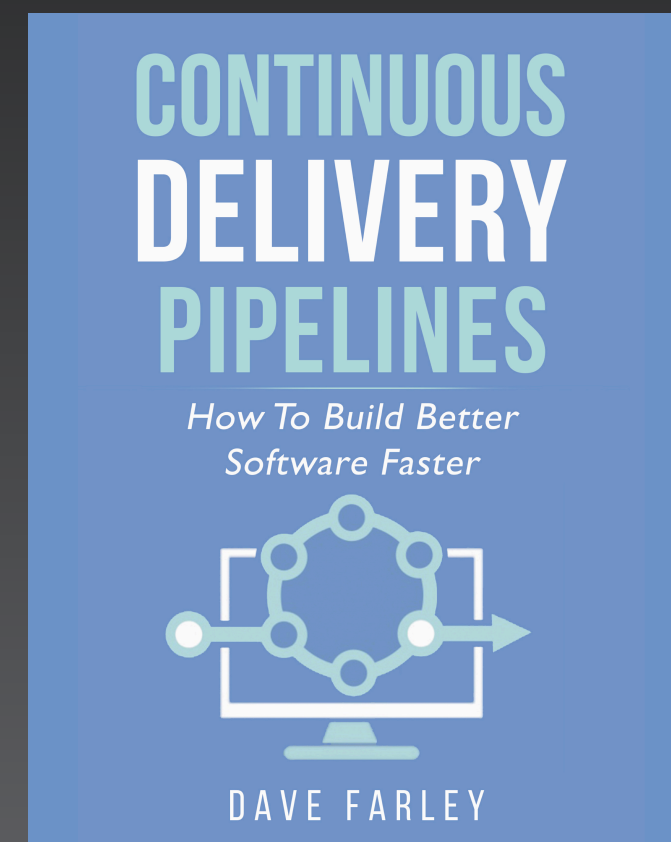
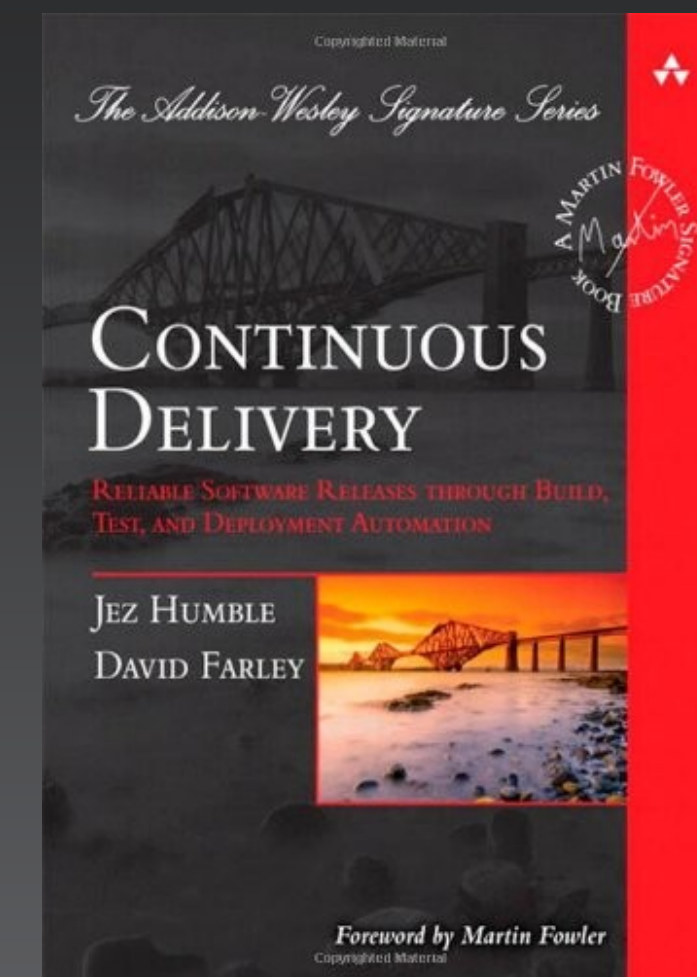
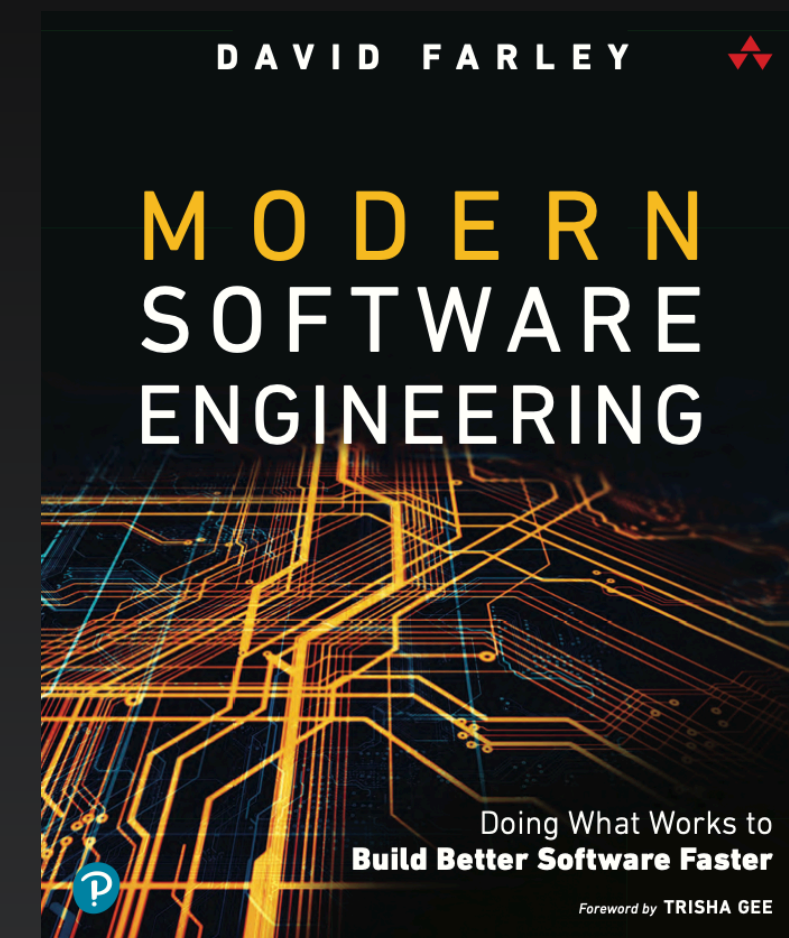
@davefarley77



<https://bit.ly/CDonYT>

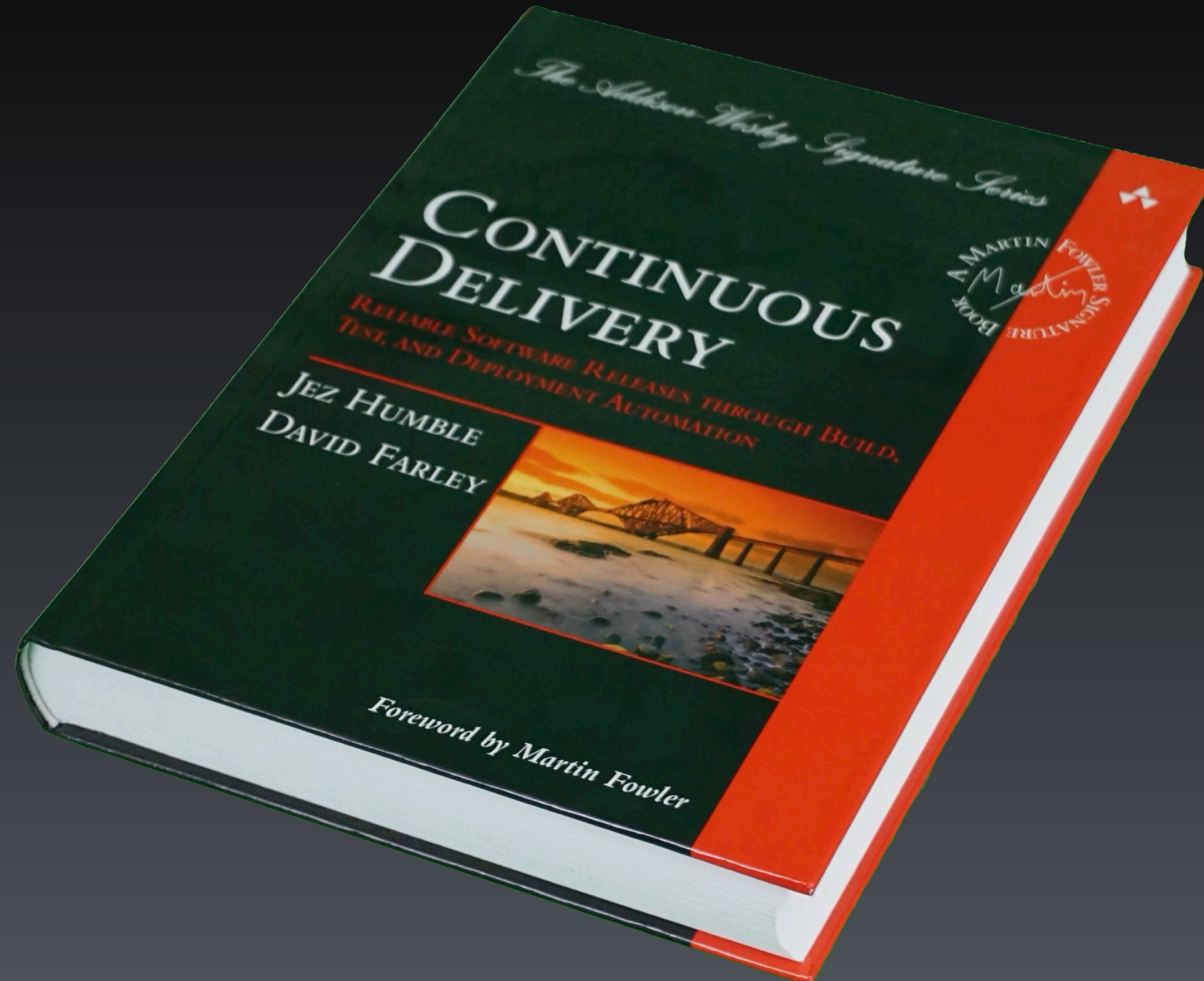


<http://www.continuous-delivery.co.uk>



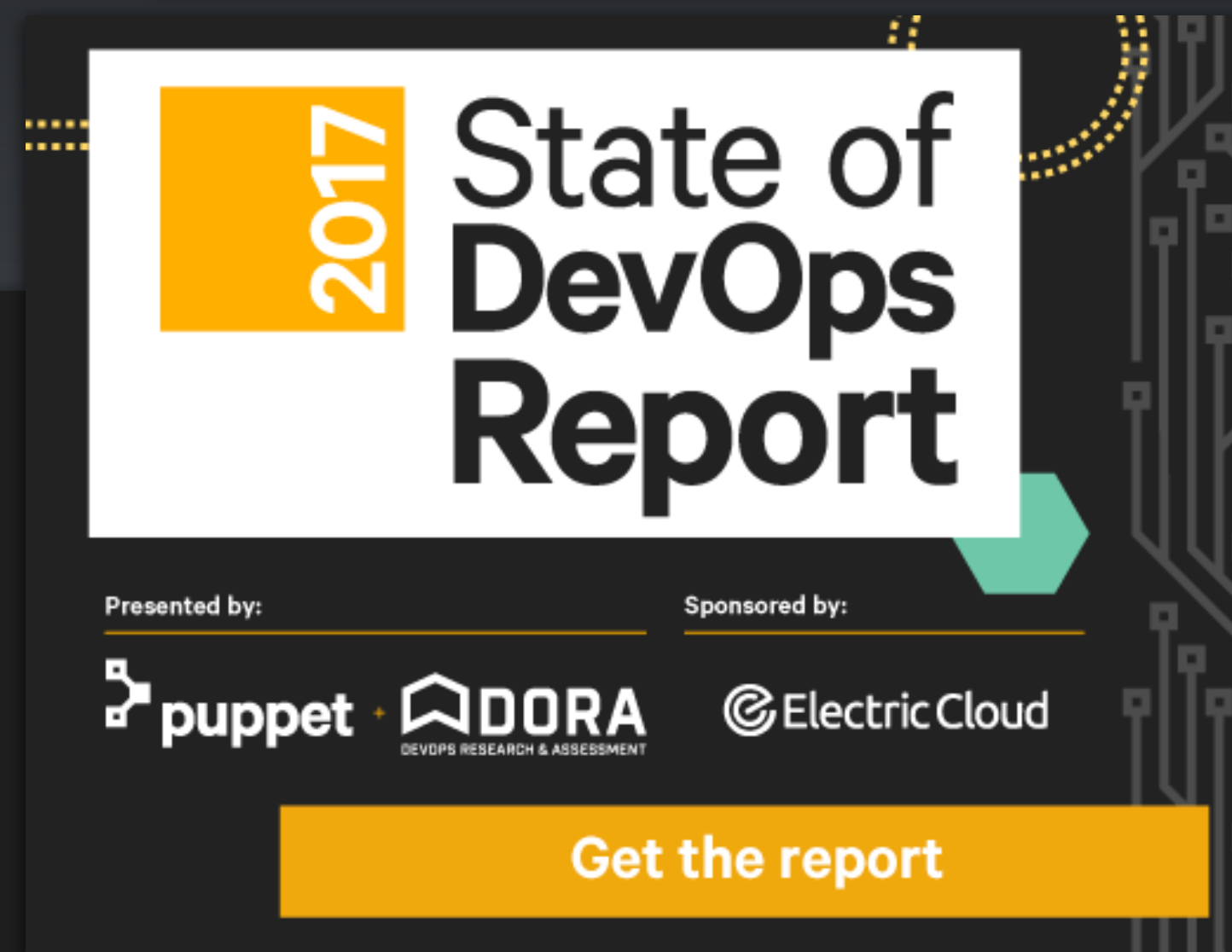
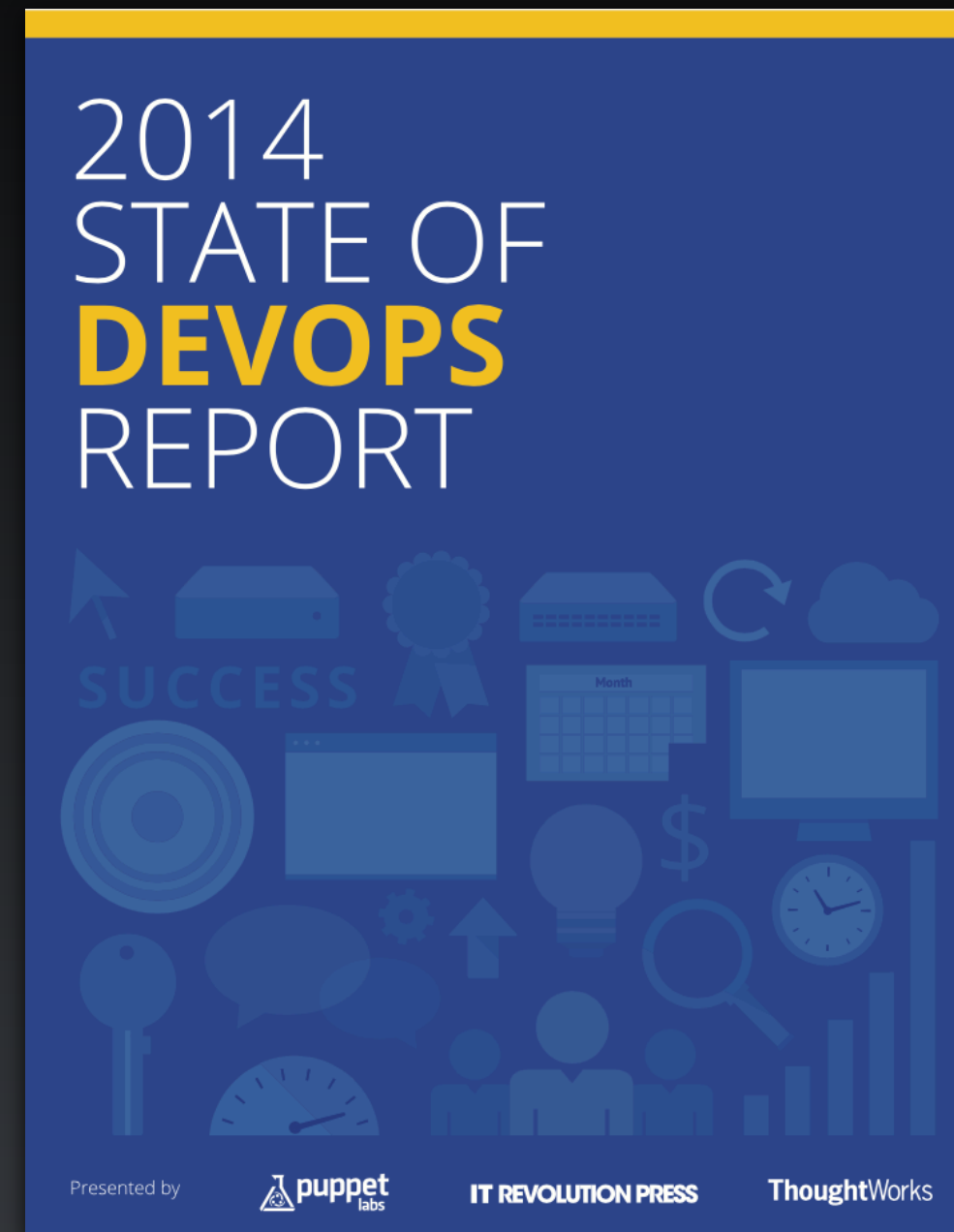
**What do you think of
when you hear the phrase**

What do you think of
when you hear the phrase
"Continuous Delivery"?

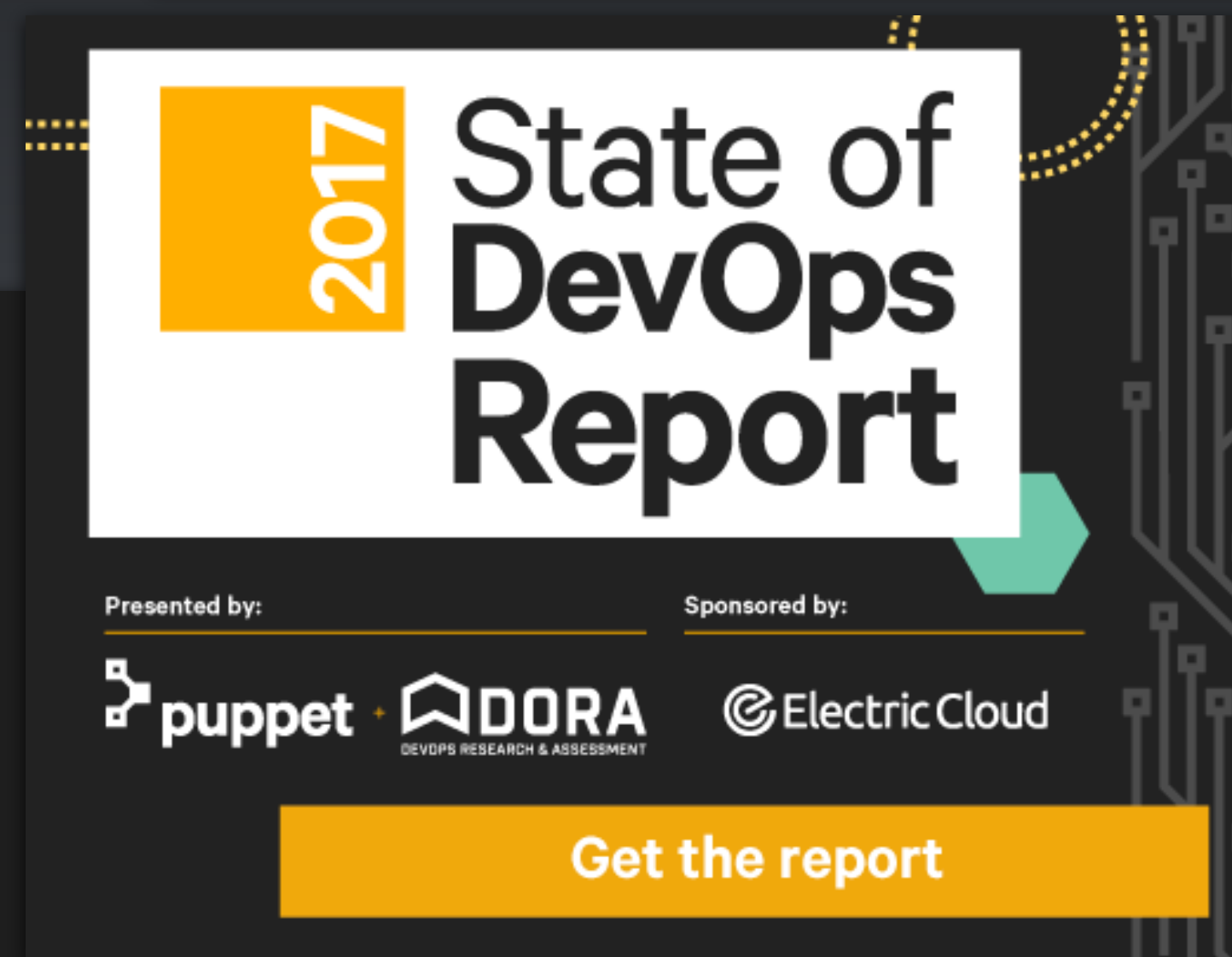
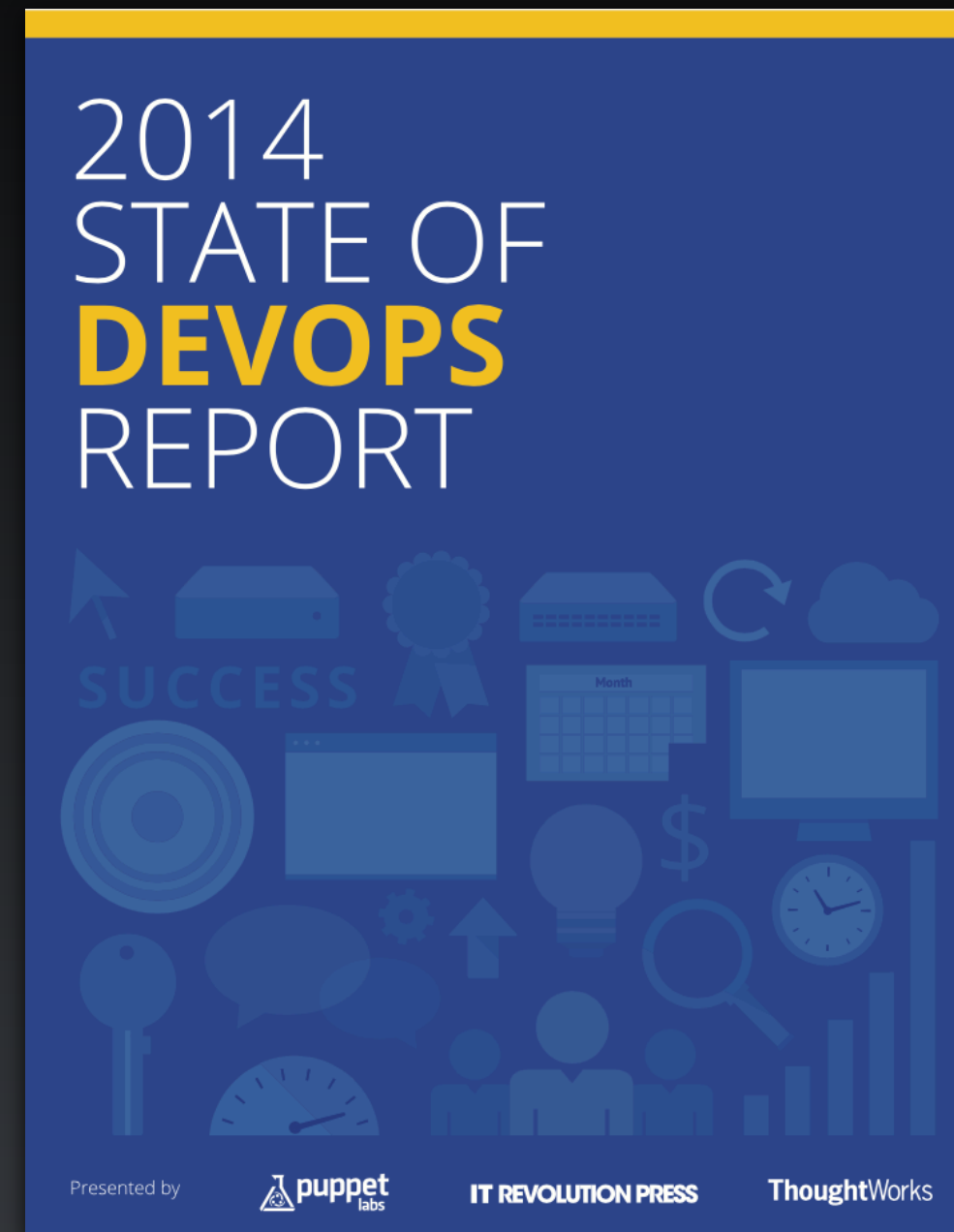


The State of DevOps Reports

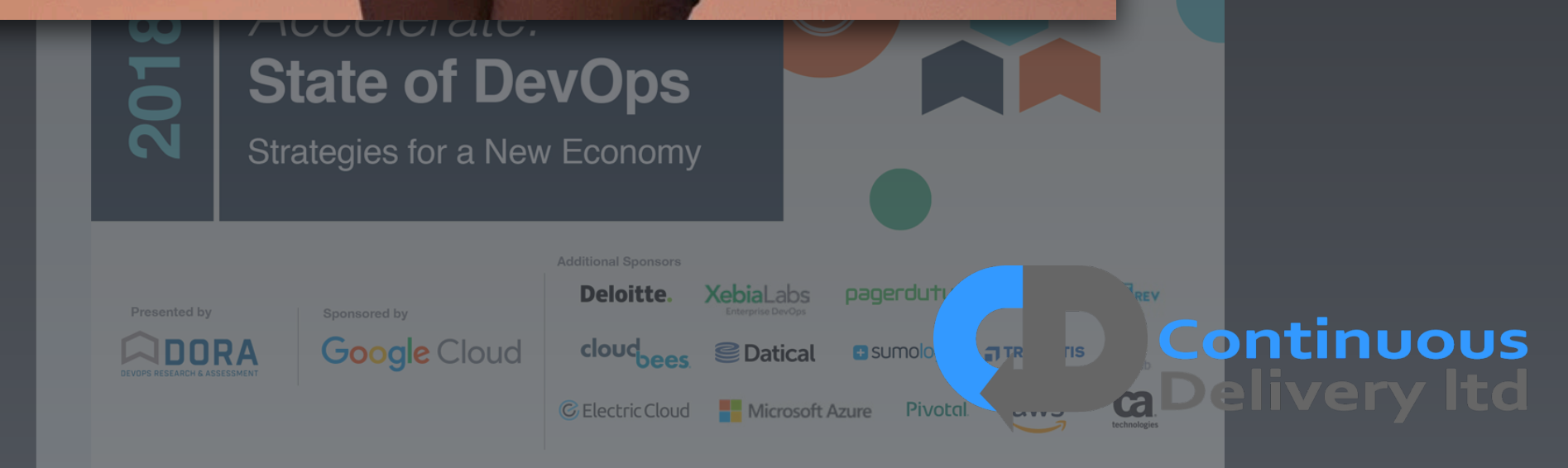
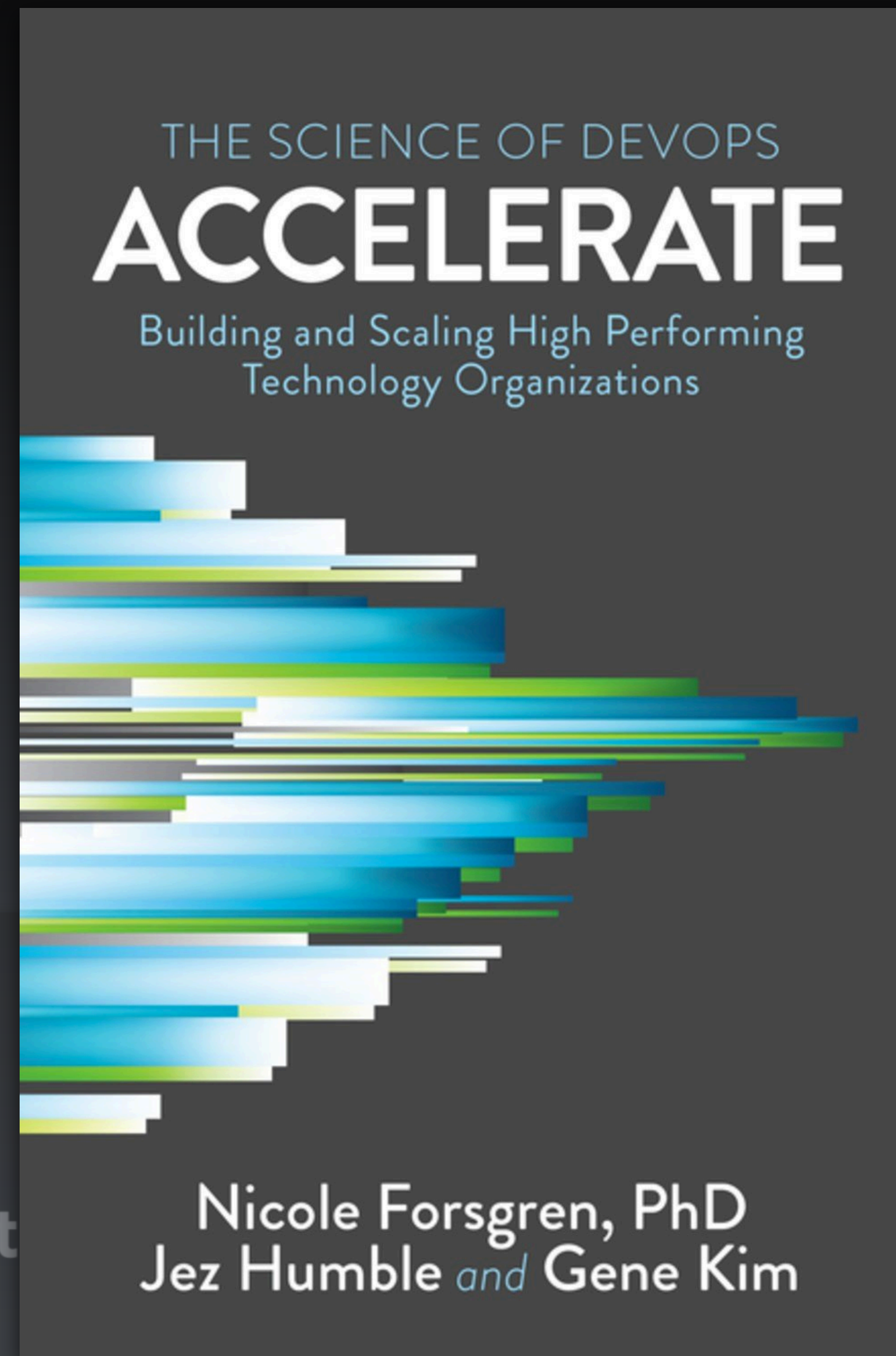
The State of DevOps Reports



The State of DevOps Reports



The State of DevOps Reports



Throughput = Lead Time & Frequency

Efficiency

Throughput

=

Lead Time

&

Frequency

Speed

Throughput = Lead Time & Frequency

Speed

Throughput = Lead Time & Frequency

Stability = Change Failure Rate & Recovery Failure Time

Speed

Throughput

=

Lead Time

&

Frequency

Quality

Stability

=

Change
Failure Rate

&

Recovery
Failure Time

Speed

Throughput = Lead Time & Frequency

Quality

Stability = Change Failure Rate & Recovery Failure Time

ELITE PERFORMERS

Comparing the elite group against the low performers, we find that elite performers have...



208
TIMES MORE
frequent code deployments

106
TIMES FASTER
lead time from
commit to deploy



2,604
TIMES FASTER
time to recover from incidents

7
TIMES LOWER
change failure rate
(changes are $\frac{1}{7}$ as likely to fail)



Throughput Stability

Speed

Throughput

Quality

Stability

Lead time

Change failure rate

ELITE PERFORMERS

Comparing the elite group against the low performers, we find that elite performers have

Speed



208
TIMES MORE
frequent code deployments



106
TIMES FASTER
lead time from
commit to deploy



2,604
TIMES FASTER
time to recover from incidents



7
TIMES LOWER
change failure rate
(changes are $\frac{1}{7}$ as likely to fail)

Throughput Stability

ELITE PERFORMERS

Comparing the elite group against the low performers, we find that elite performers have

Speed



208

TIMES MORE

frequent code deployments

106

TIMES FASTER

lead time from
commit to deploy



Quality



2,604

TIMES FASTER

time to recover from incidents

7

TIMES LOWER

change failure rate
(changes are $\frac{1}{7}$ as likely to fail)



Throughput Stability

ELITE PERFORMERS

Comparing the elite group against the low performers, we find that elite performers have

Speed



208

TIMES MORE

frequent code deployments

106

TIMES FASTER

lead time from
commit to deploy



Quality



2,604

TIMES FASTER

time to recover from incidents

7

TIMES LOWER

change failure rate
(changes are $\frac{1}{7}$ as likely to fail)



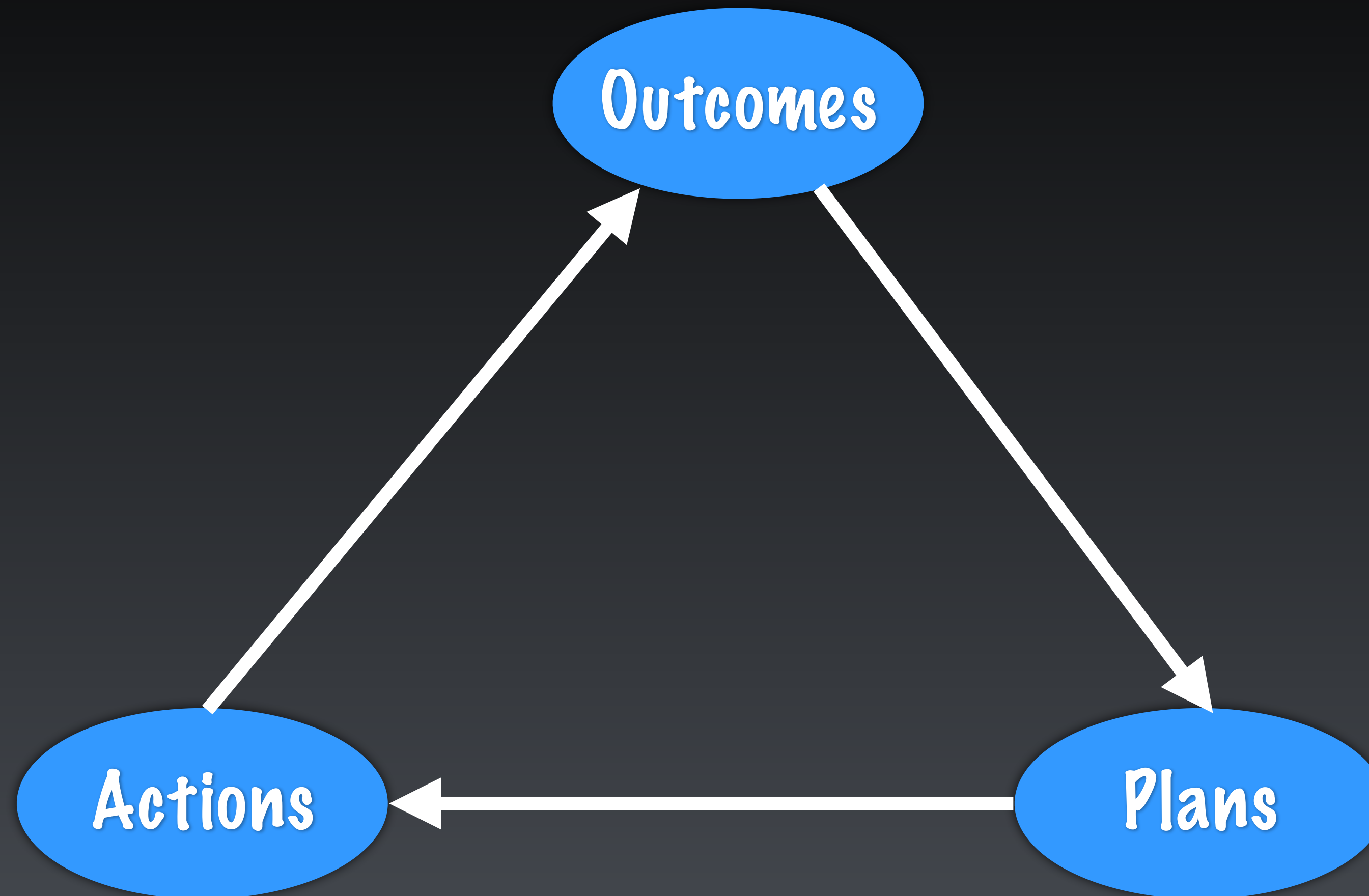
Throughput Stability

Speed & Quality

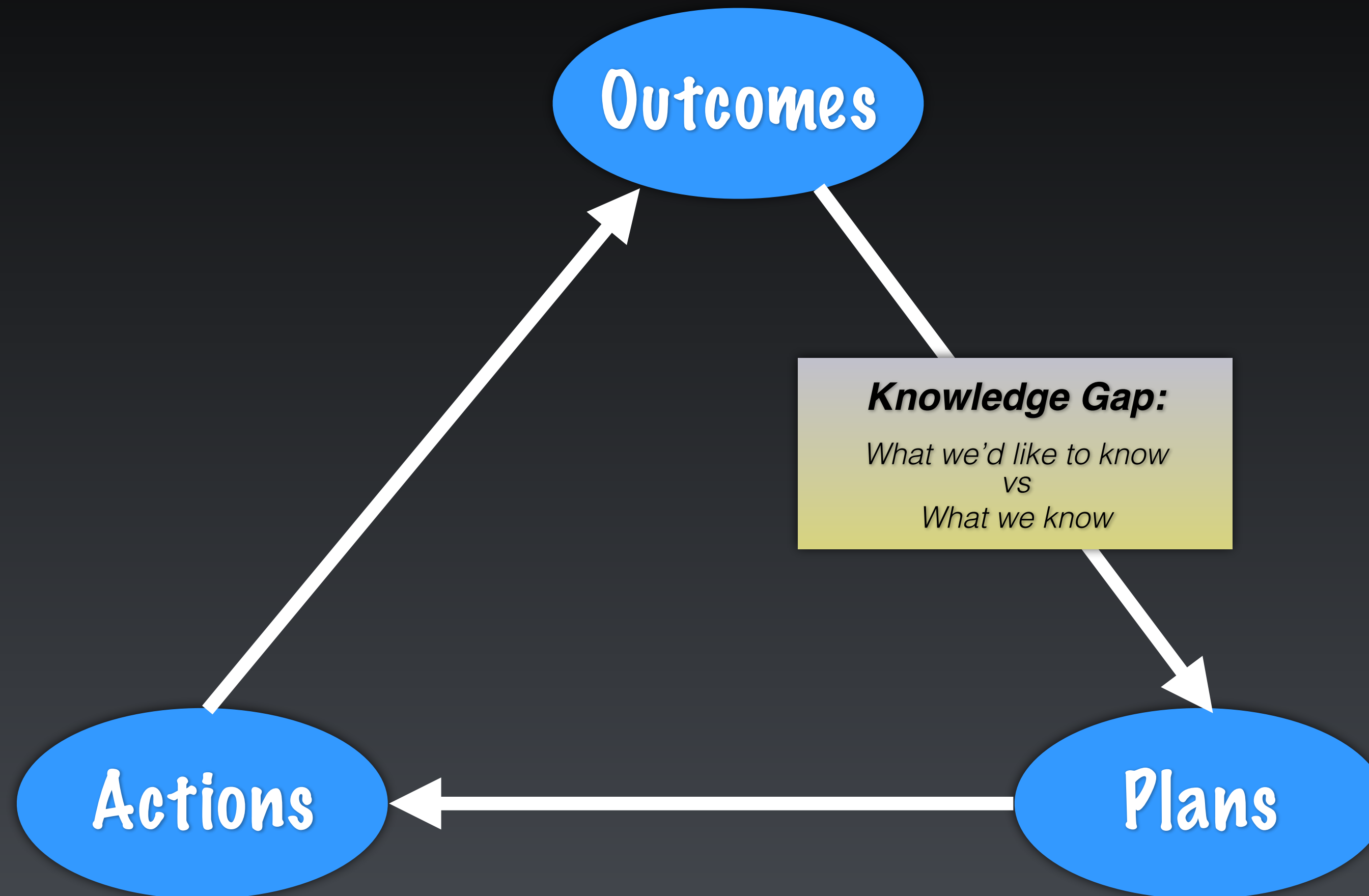
**There is No Trade-Off
Between
Speed & Quality!!!**

The Nature of the Problem

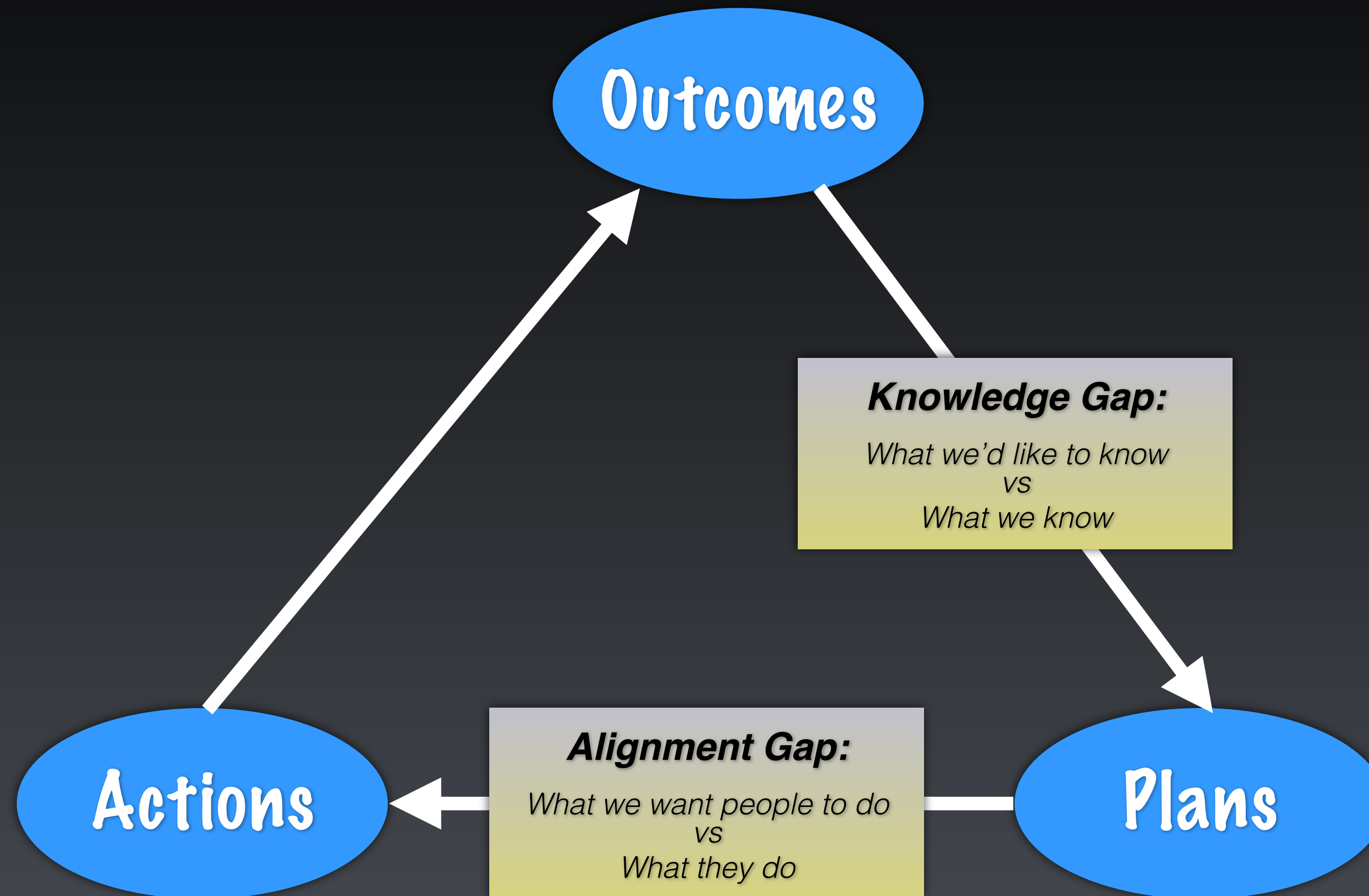
The Nature of the Problem



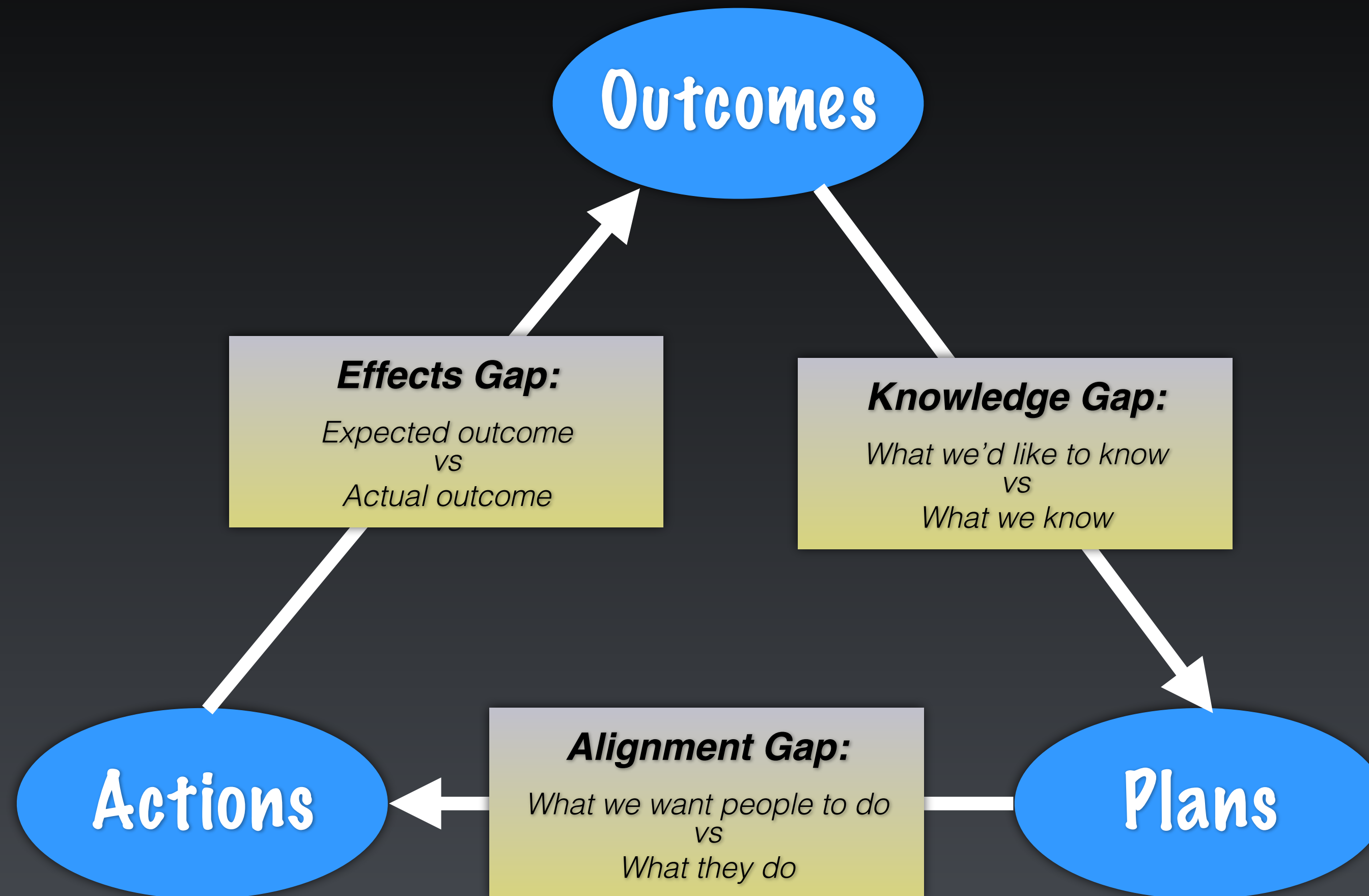
The Nature of the Problem



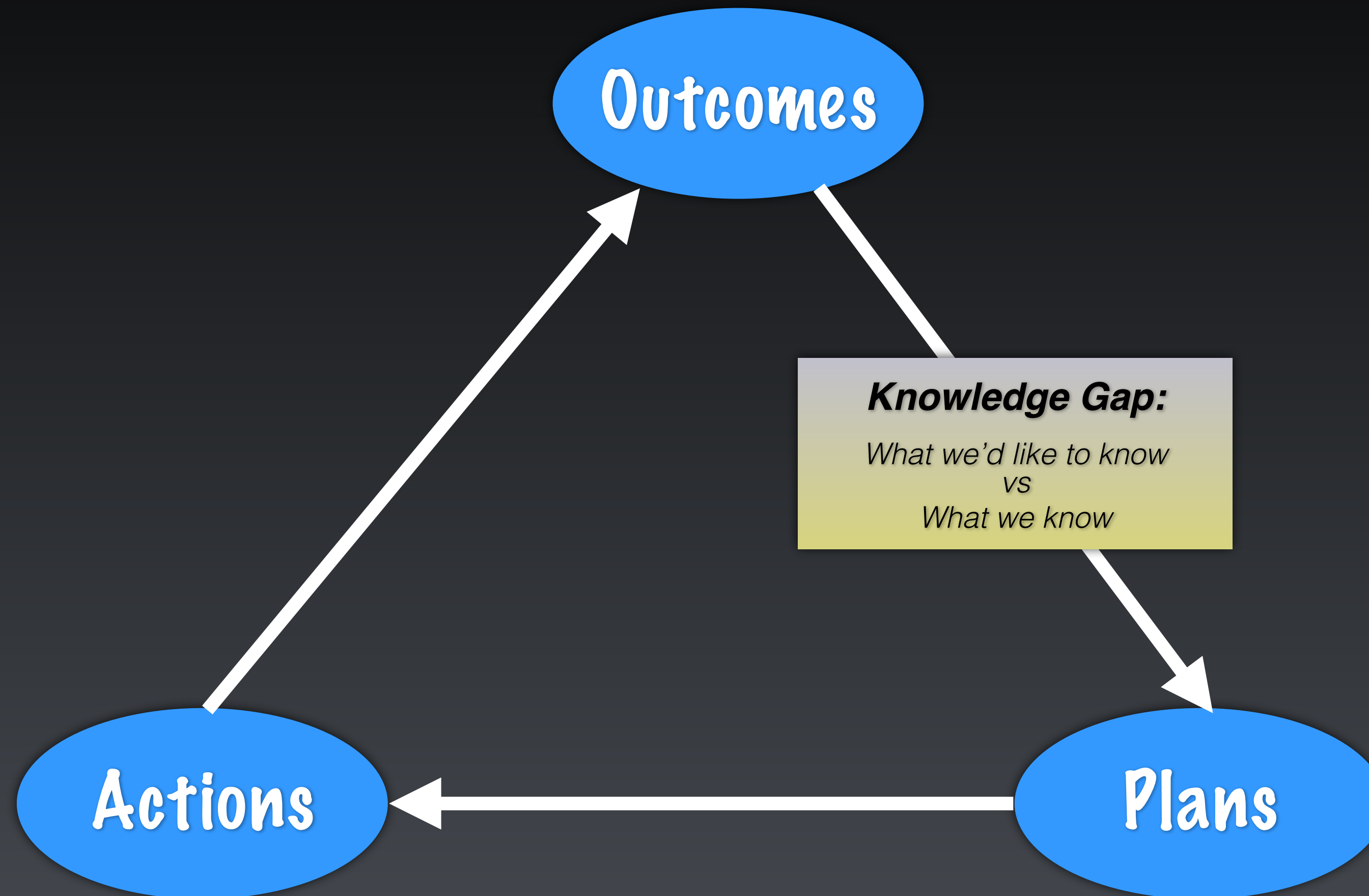
The Nature of the Problem



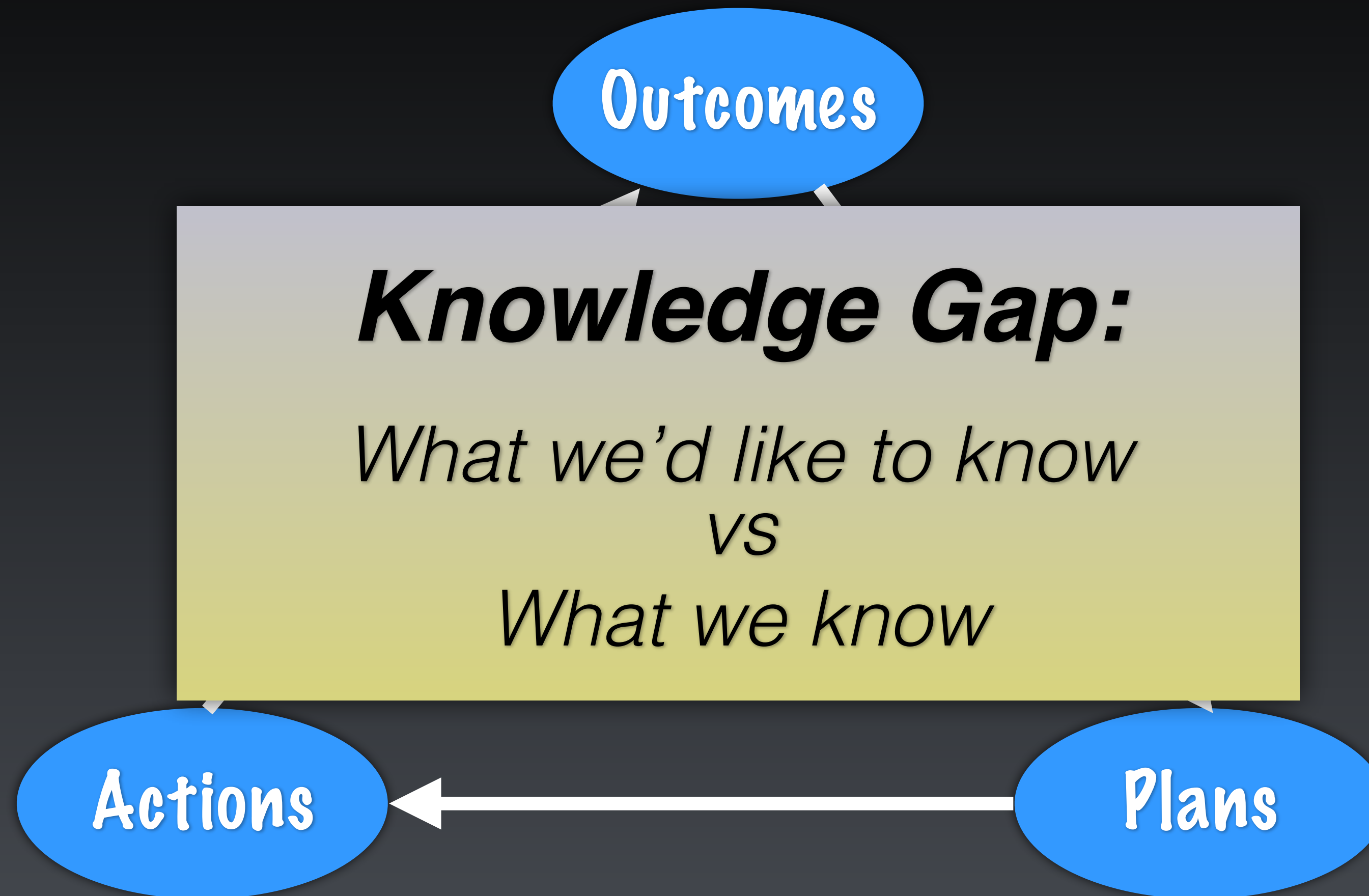
The Nature of the Problem



Classic Responses

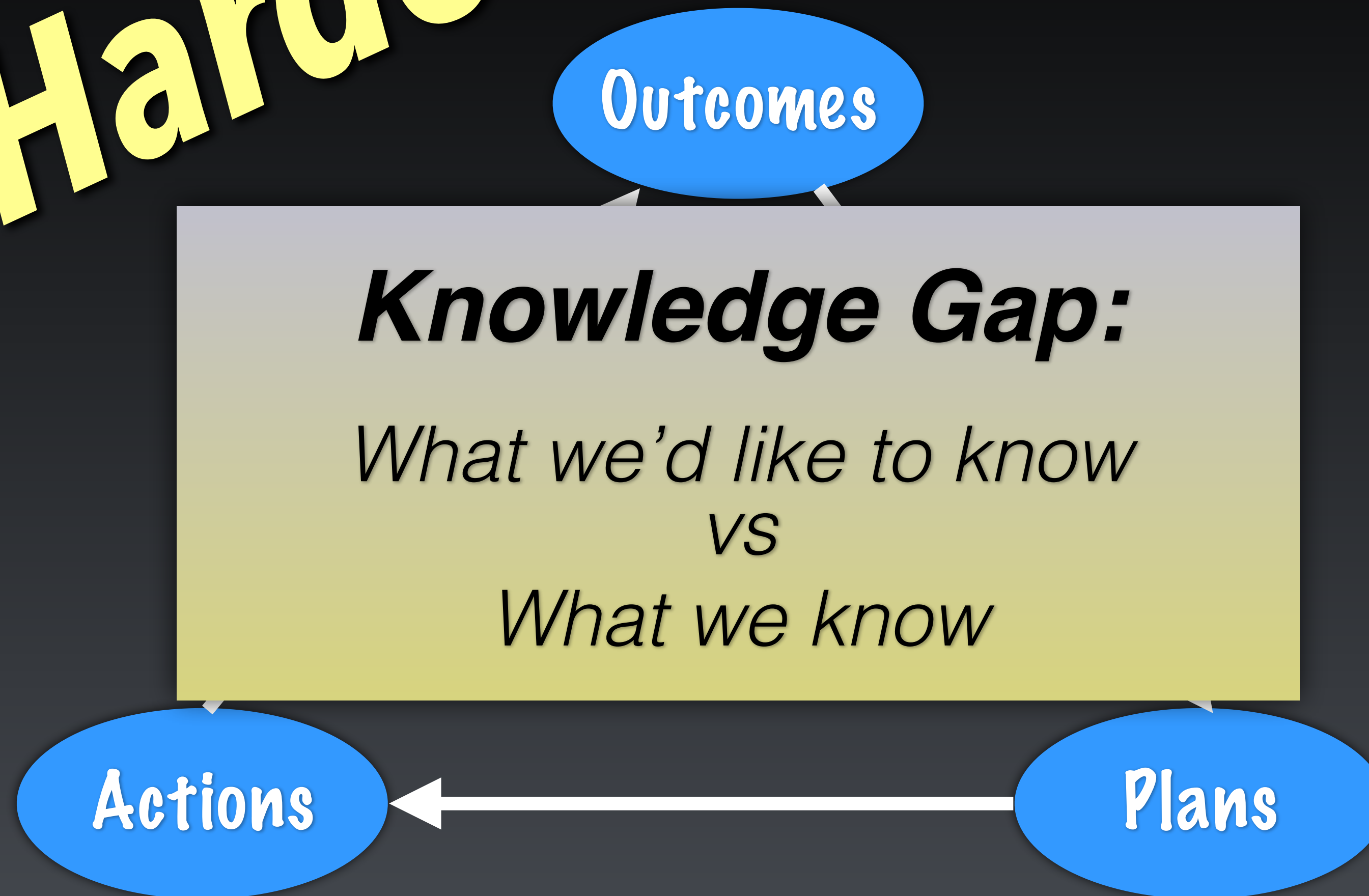


Classic Responses



Classic Responses

Plan Harder!



Classic Responses

Plan Harder!

Analyse More!

Outcomes

Knowledge Gap:

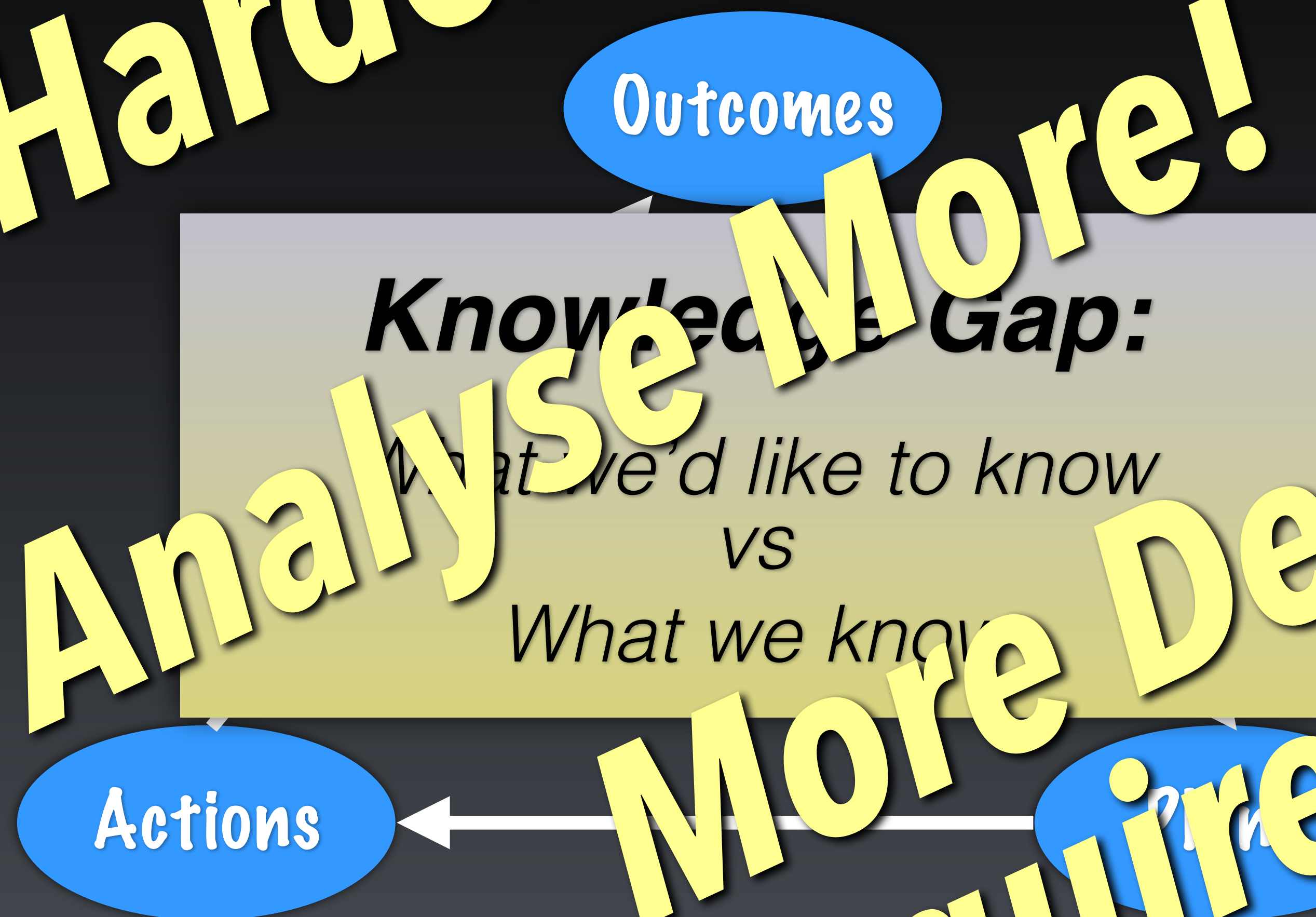
*What we'd like to know
VS*

What we know

Actions

Plans

Classic Responses



Classic Responses

Go Slower!

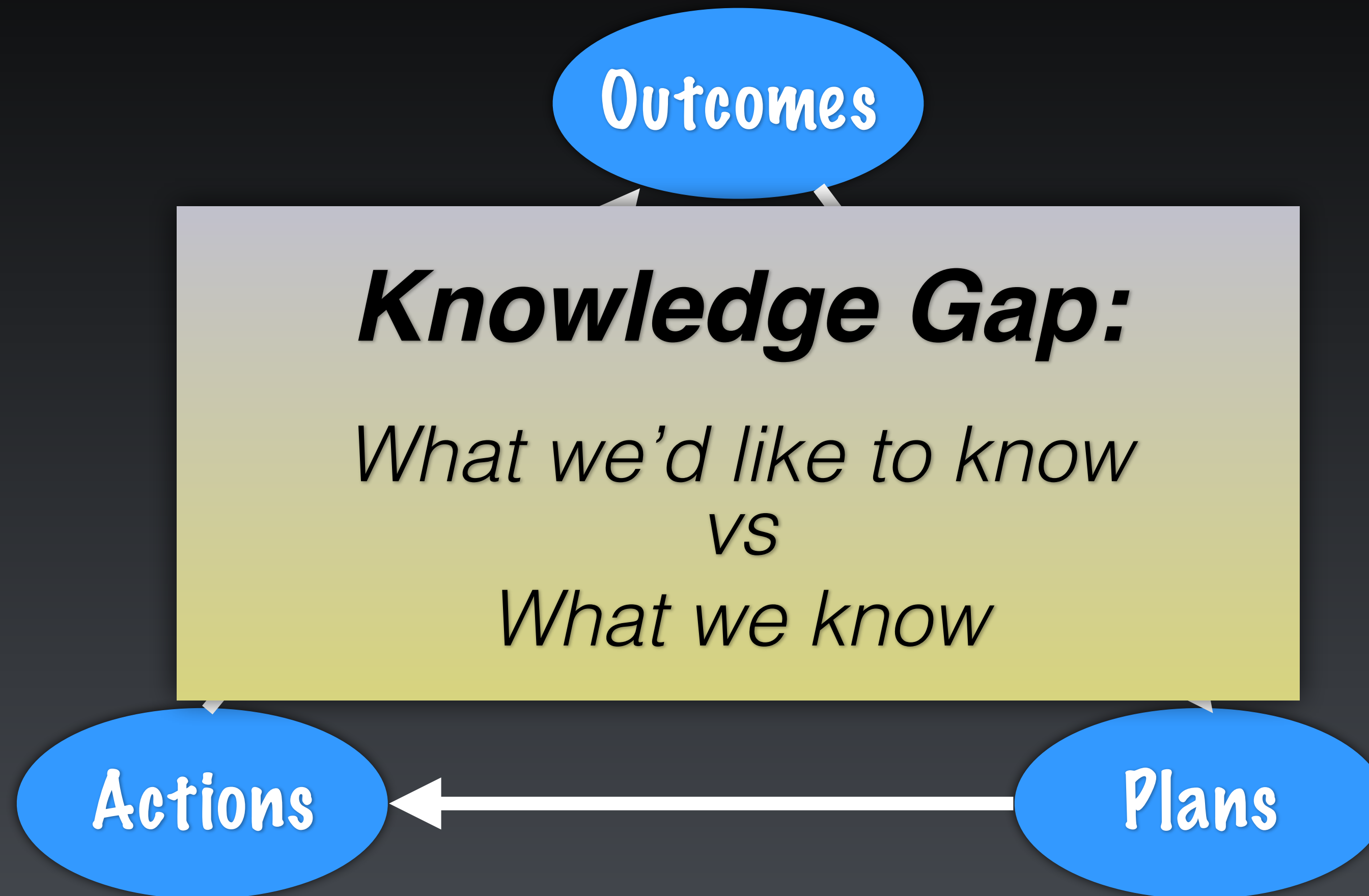
Outcomes

VS

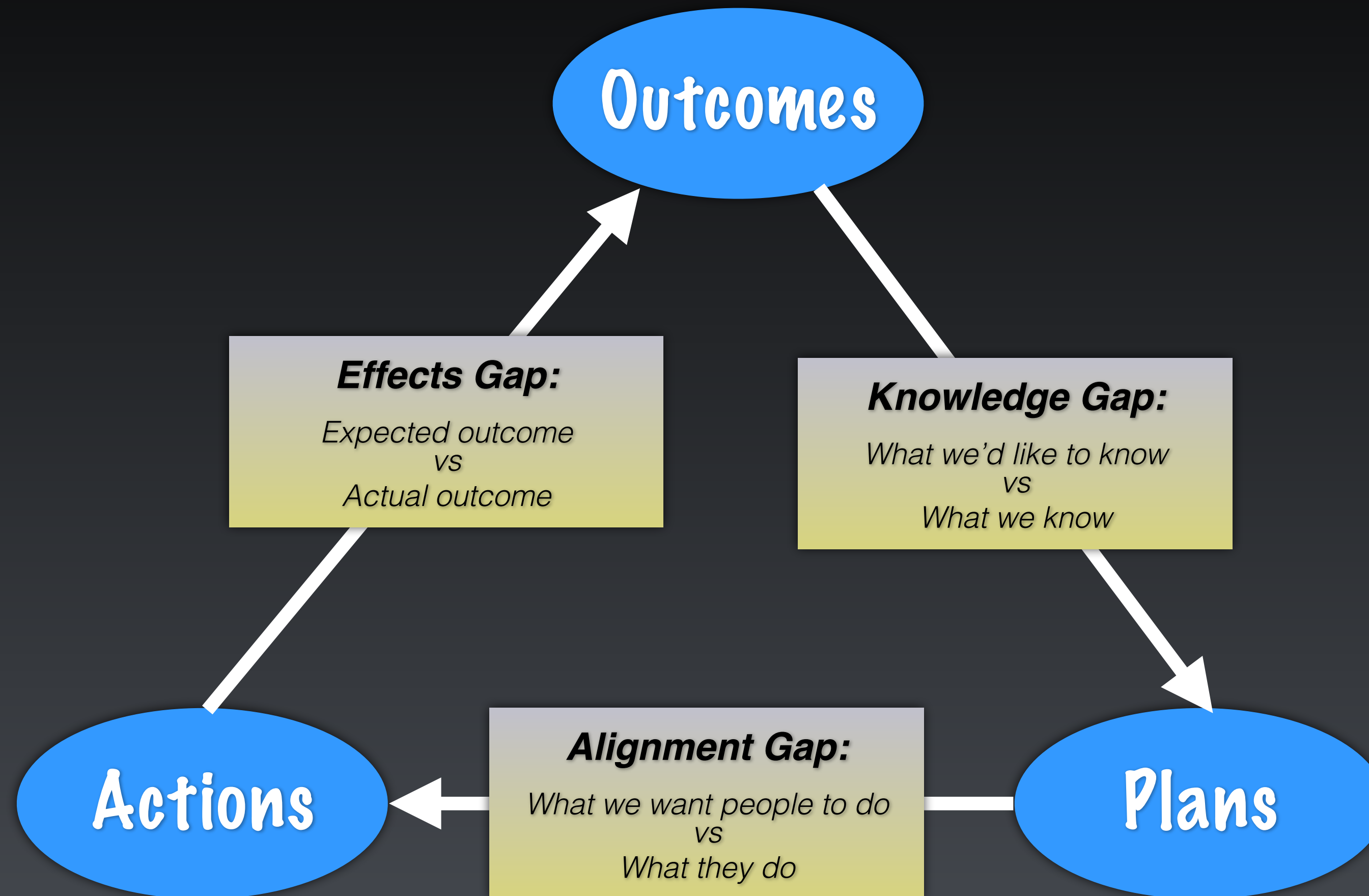
What we know

Actions

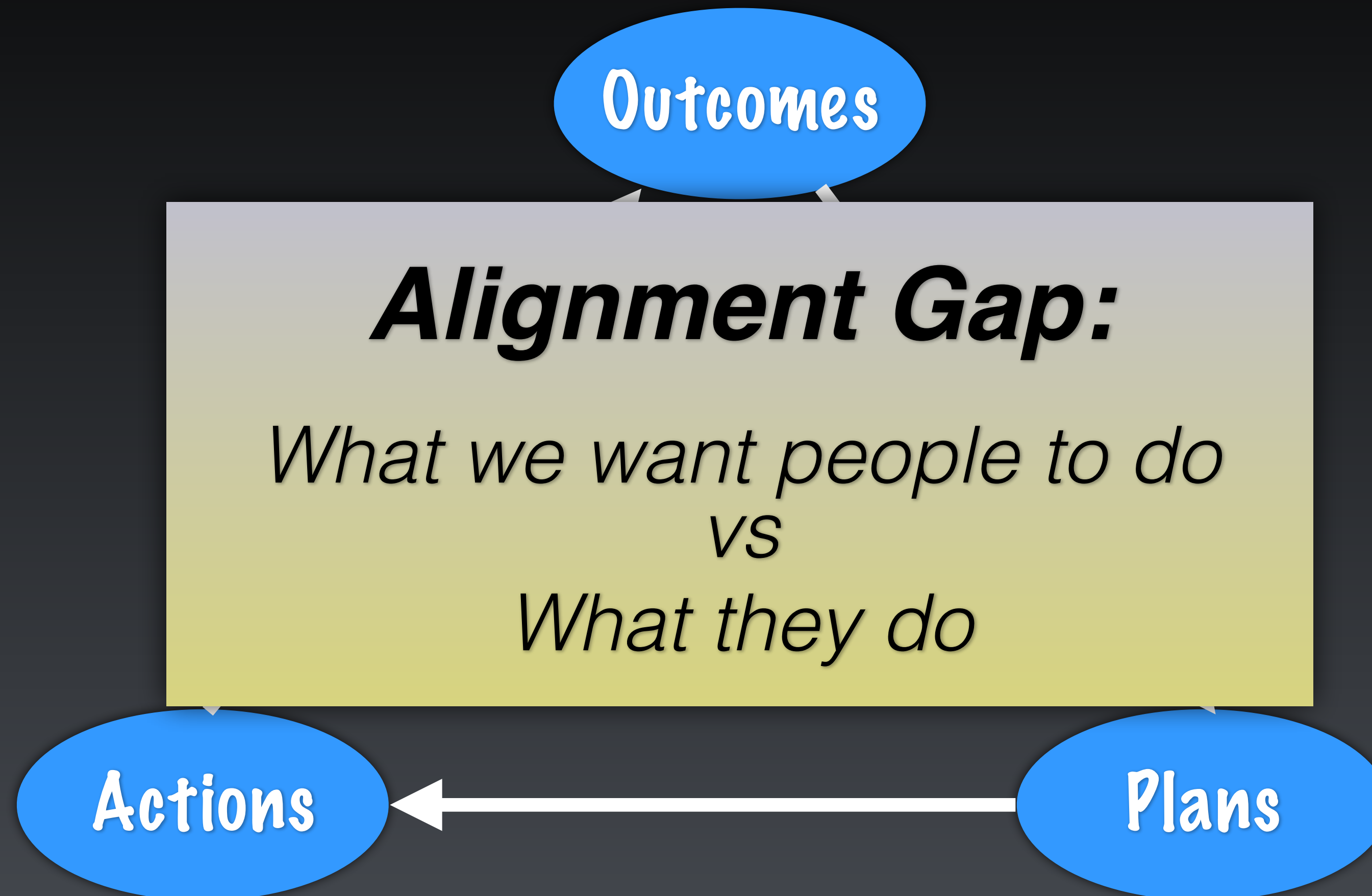
Classic Responses



Classic Responses

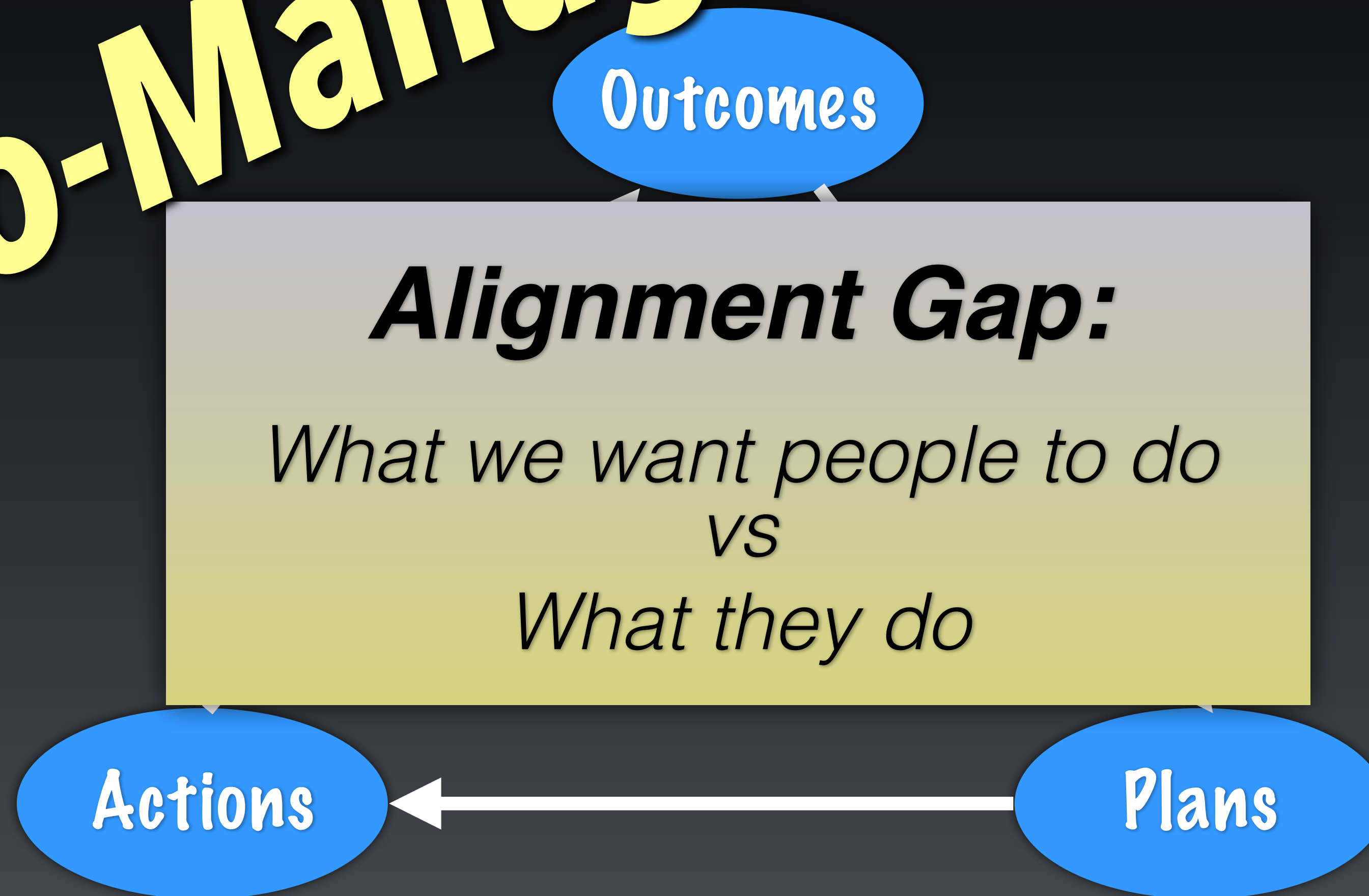


Classic Responses



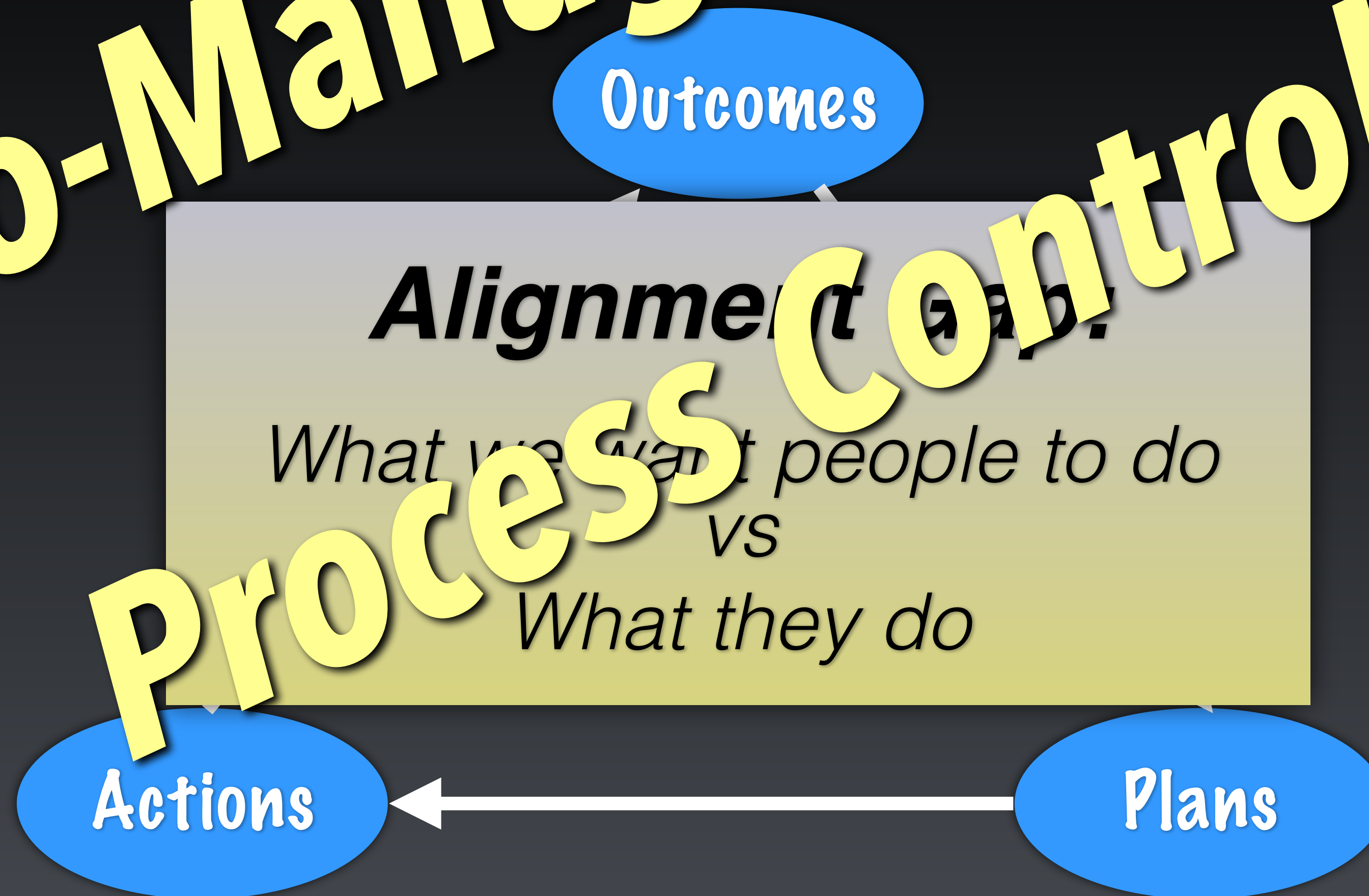
Classic Responses

Micro-Management



Classic Responses

Micro-Management
Process Control



Classic Responses

Outcomes

Alignment Gap:

What we want people to do
VS
What they do

Actions

Plans

Bureaucracy

Micro-Management

Process

Classic Responses

Outcomes

Go Slower!

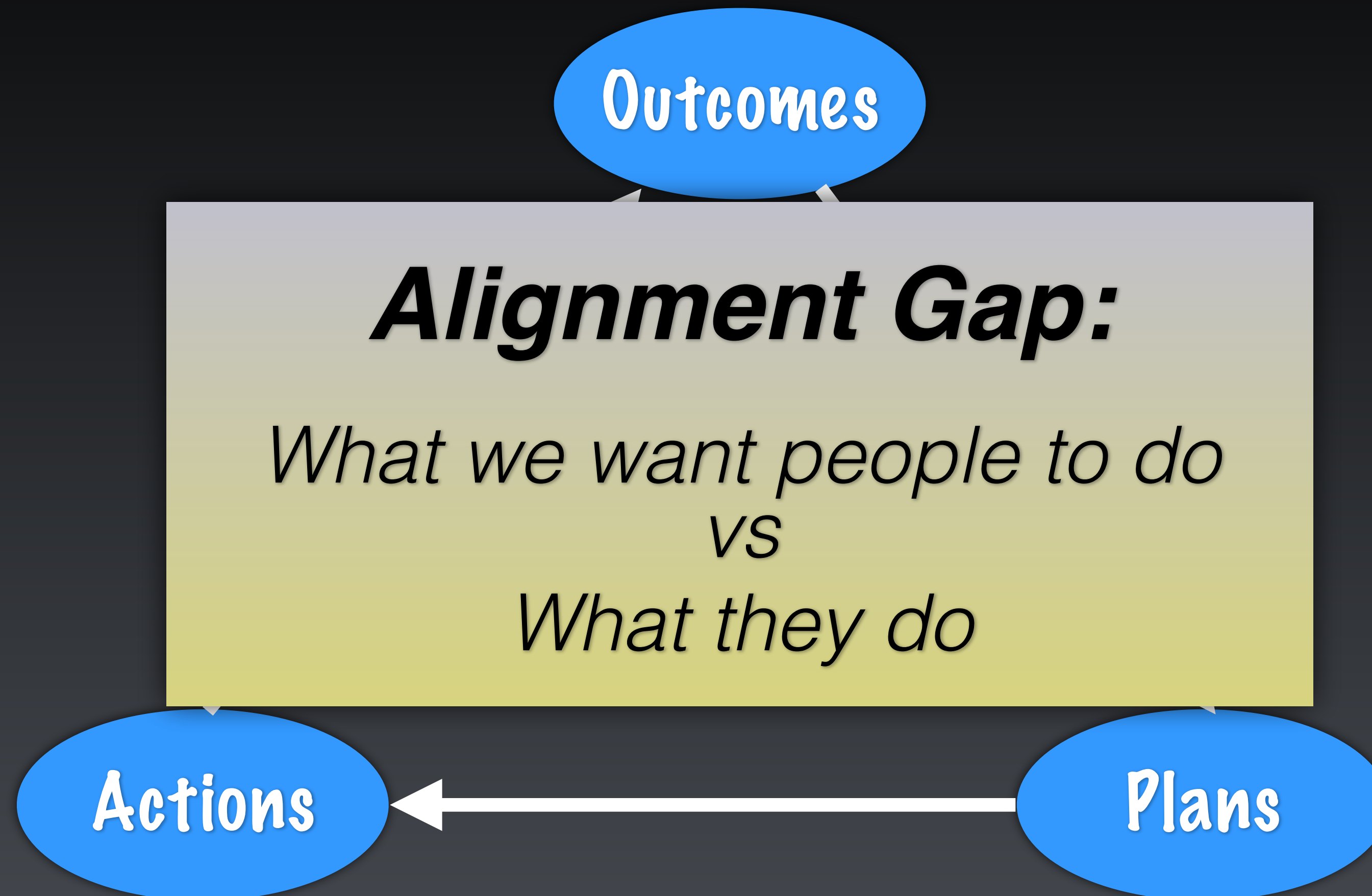
VS

What they do

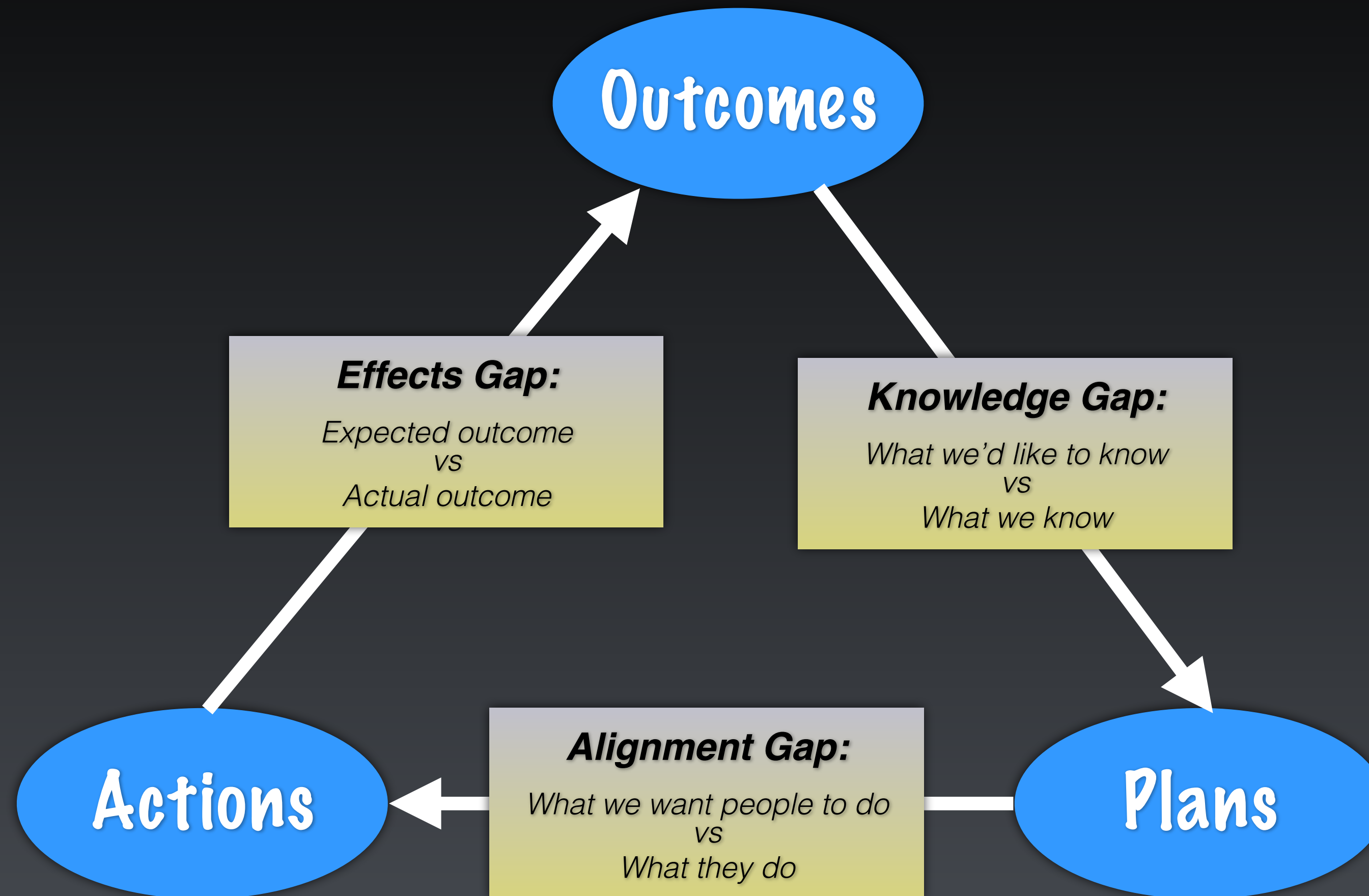
Actions

Bureaucracy

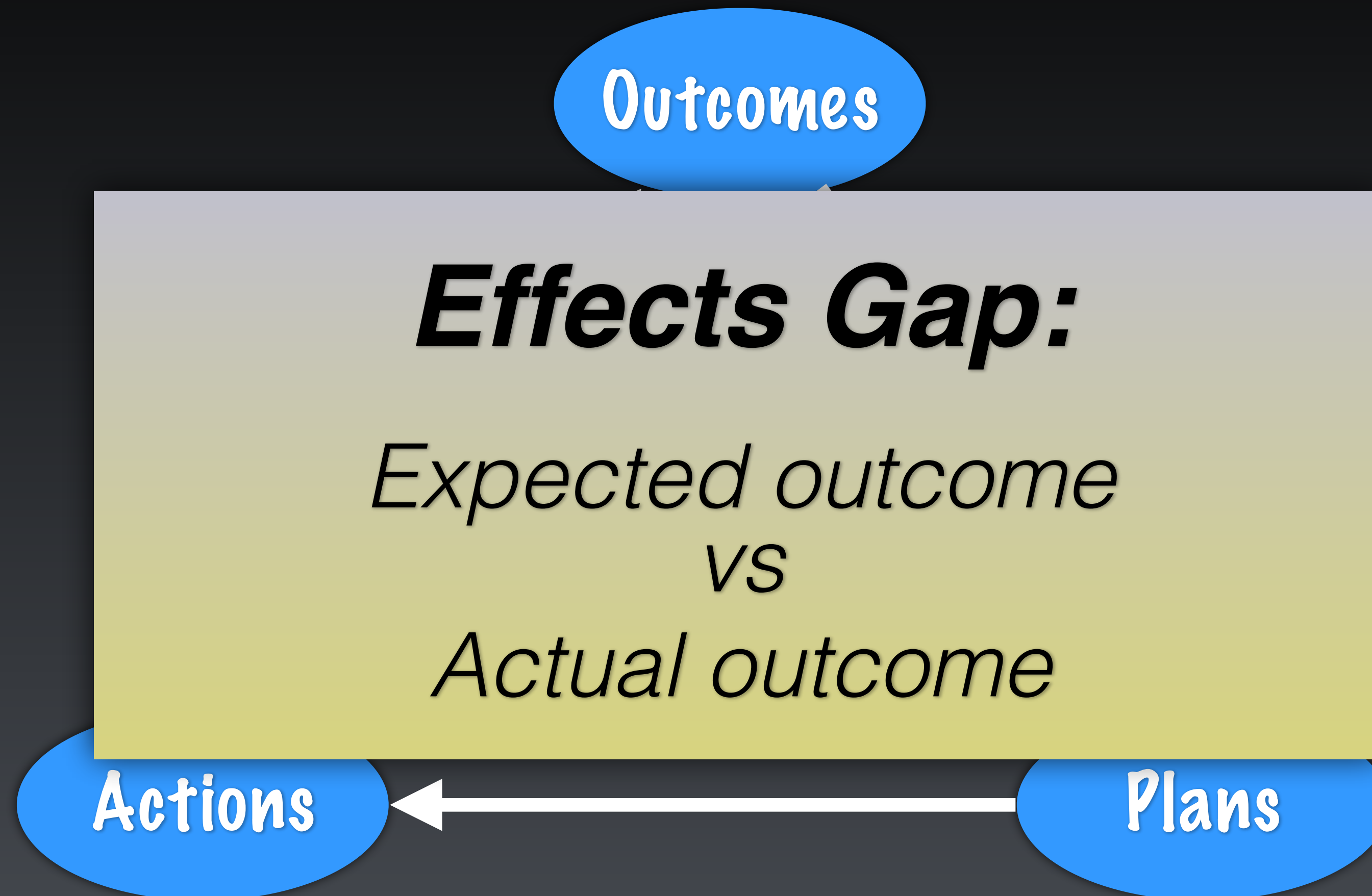
Classic Responses



Classic Responses

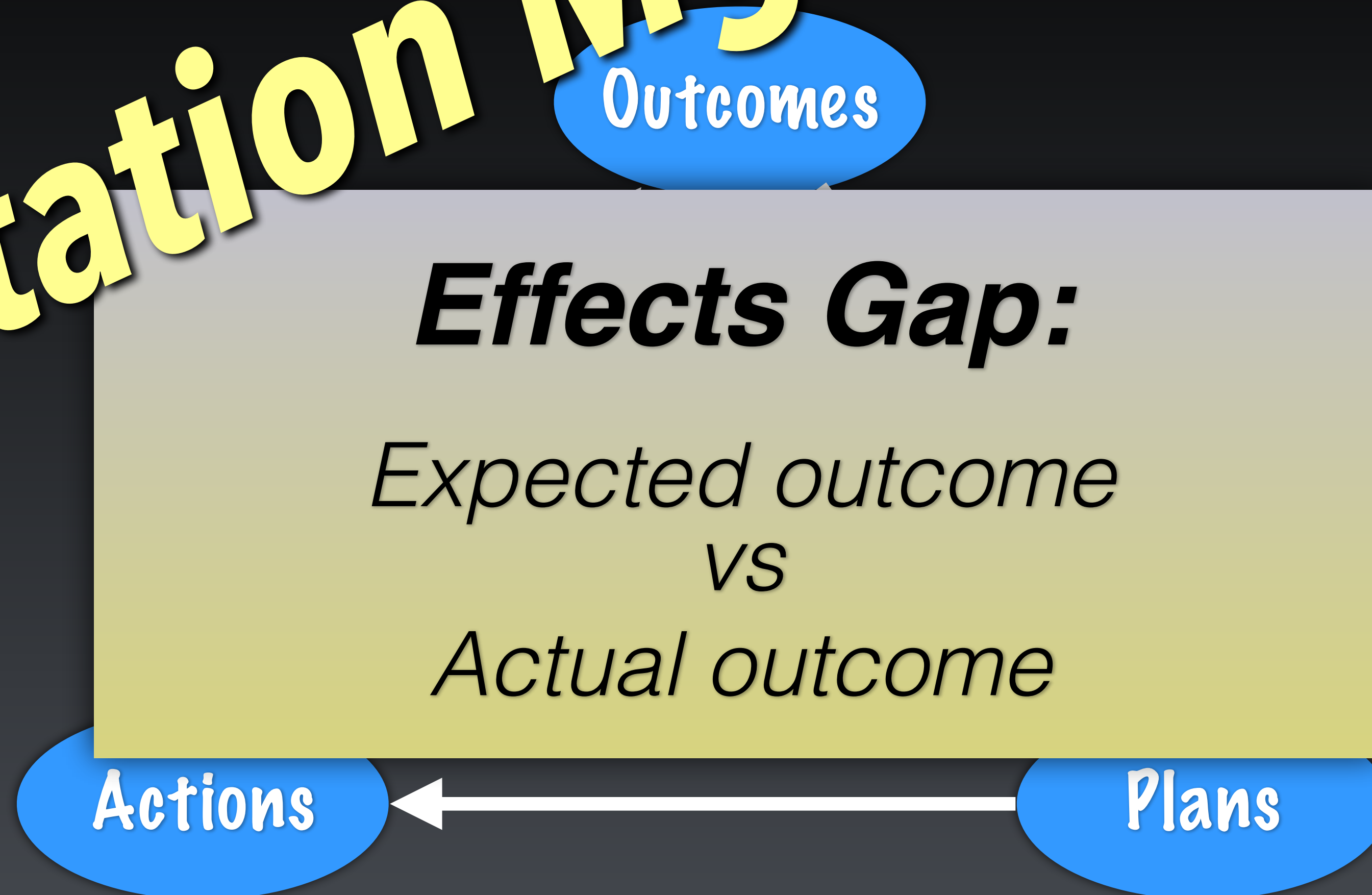


Classic Responses



Expectation Mgmt

Classic Responses



Classic Responses

Outcomes

Effects Gap:

Expected outcome

Actual outcome

Actions

Plans

Expectation Mgmt
Watermelon Status
Reporting

Classic Responses

Outcomes

Effects Gap:

Expected outcome

Actual outcome

Actions

Plans

Increase PM
Rigour

Expectation Mgmt
Watermelon Reporting

Go Slower!

Classic Responses

Outcomes

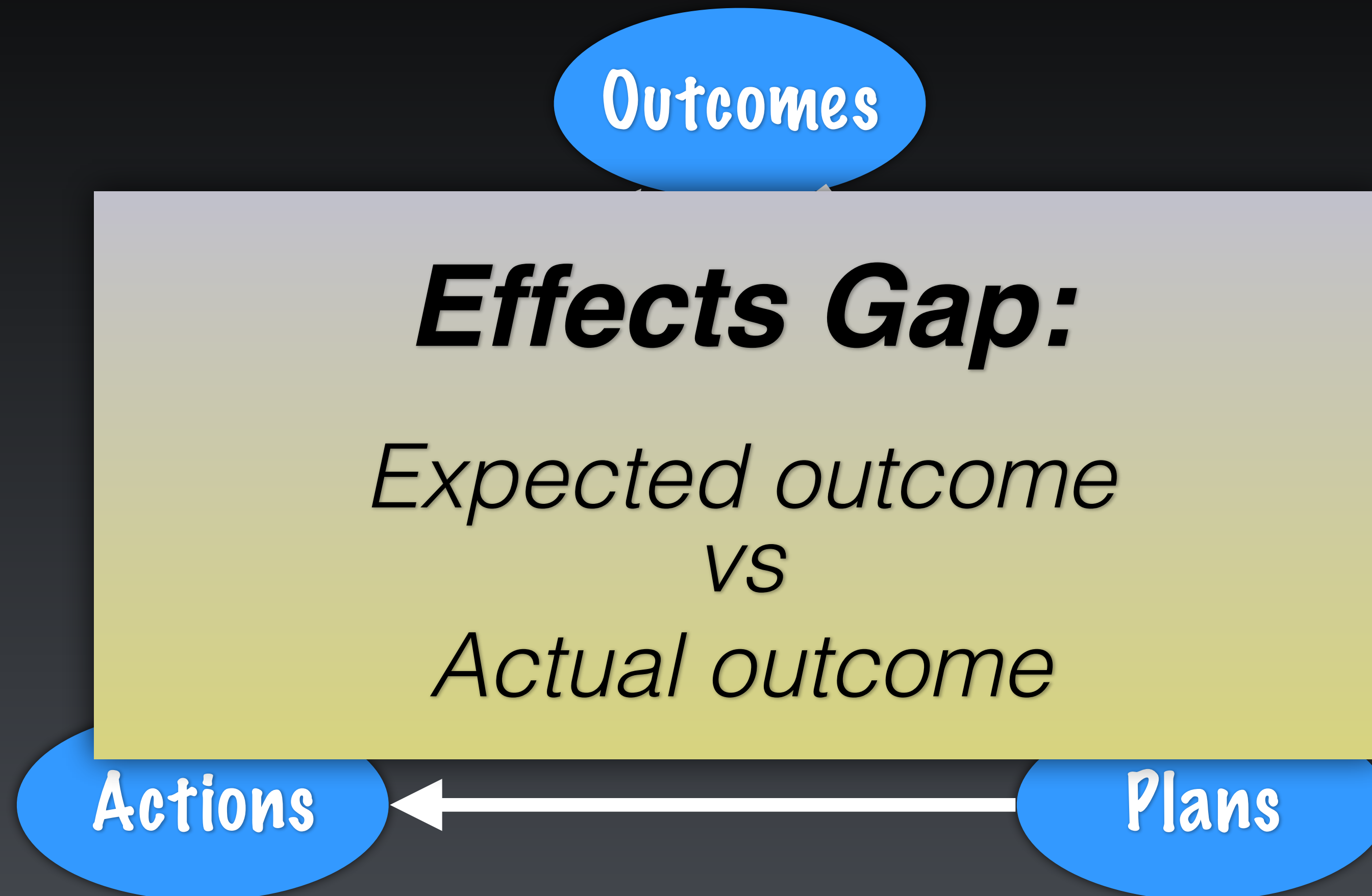
Actions

Plans



Actual outcome

Classic Responses

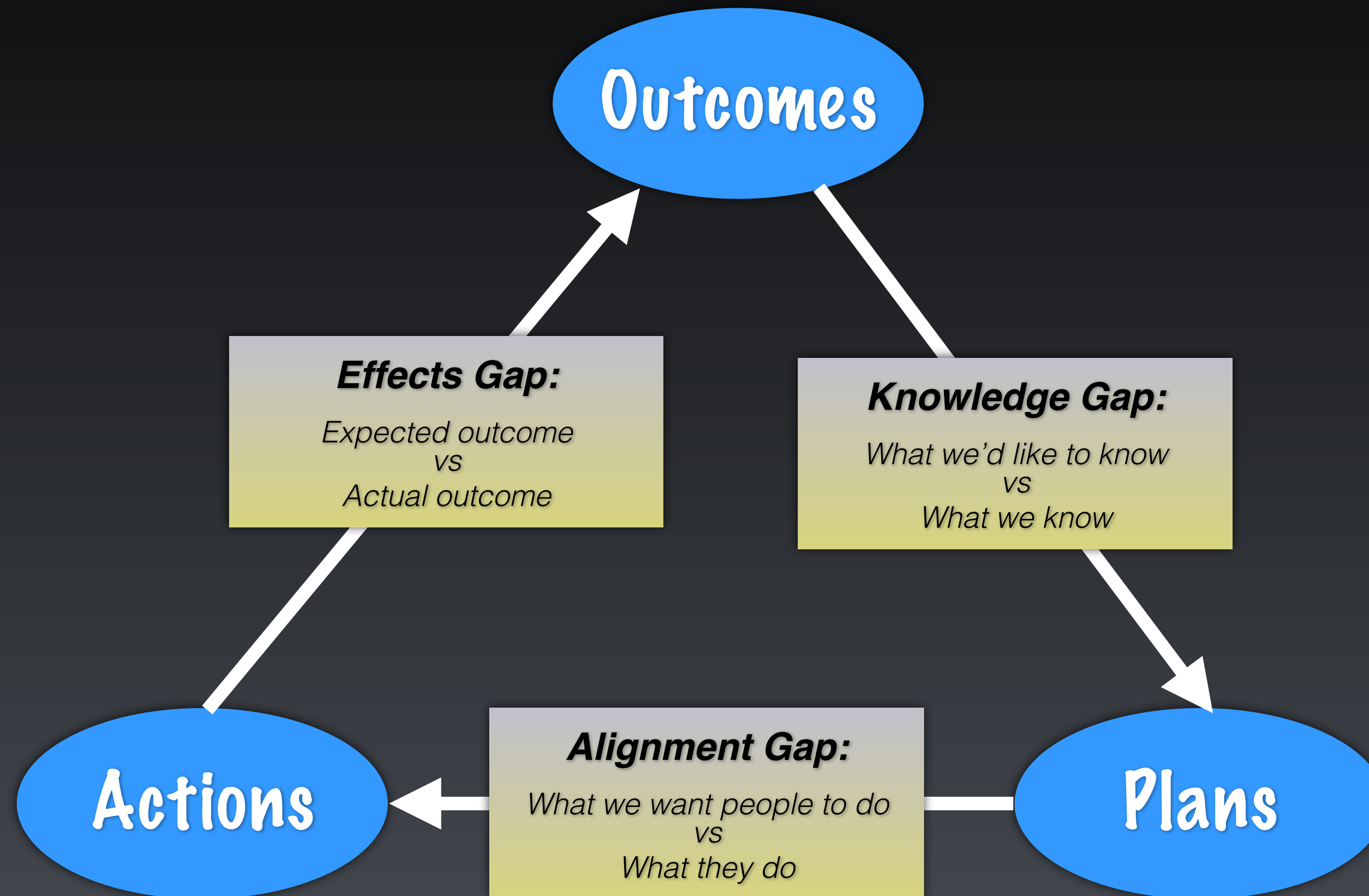


But its Worse Than That...

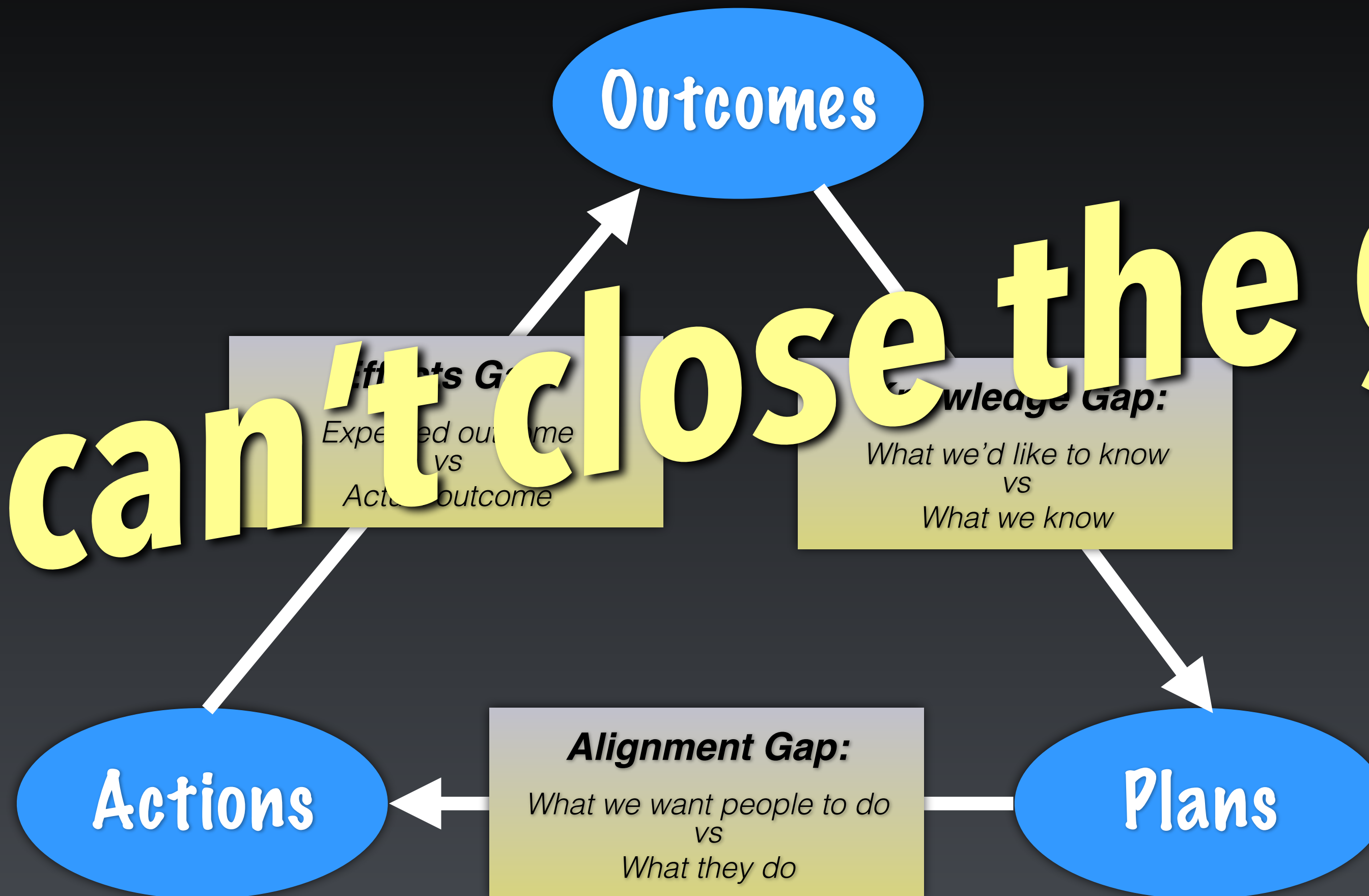
Speed
&
Quality

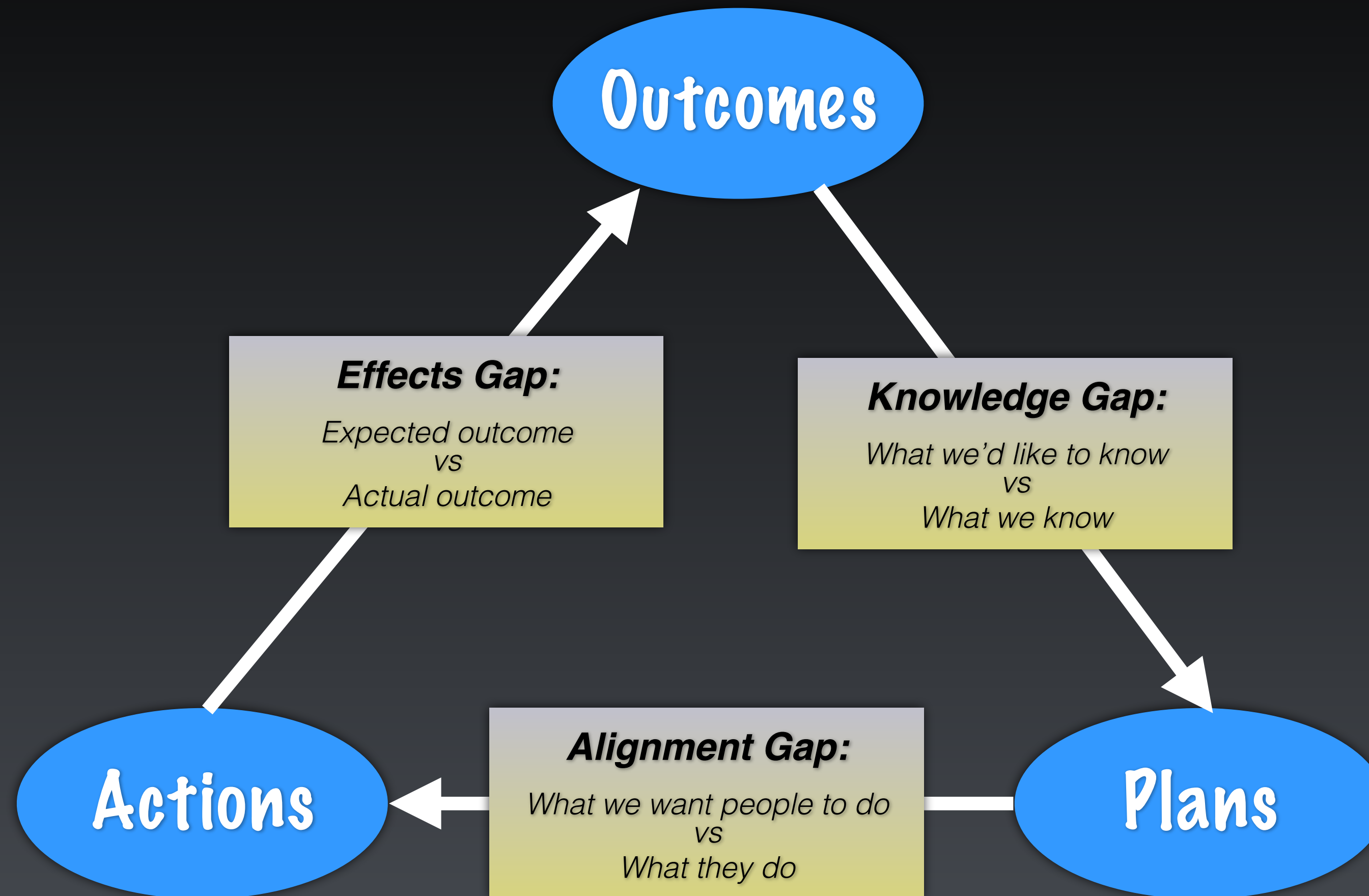
If we Go Slower...

**If we Go Slower...
We Build Worse
Software Slower!**

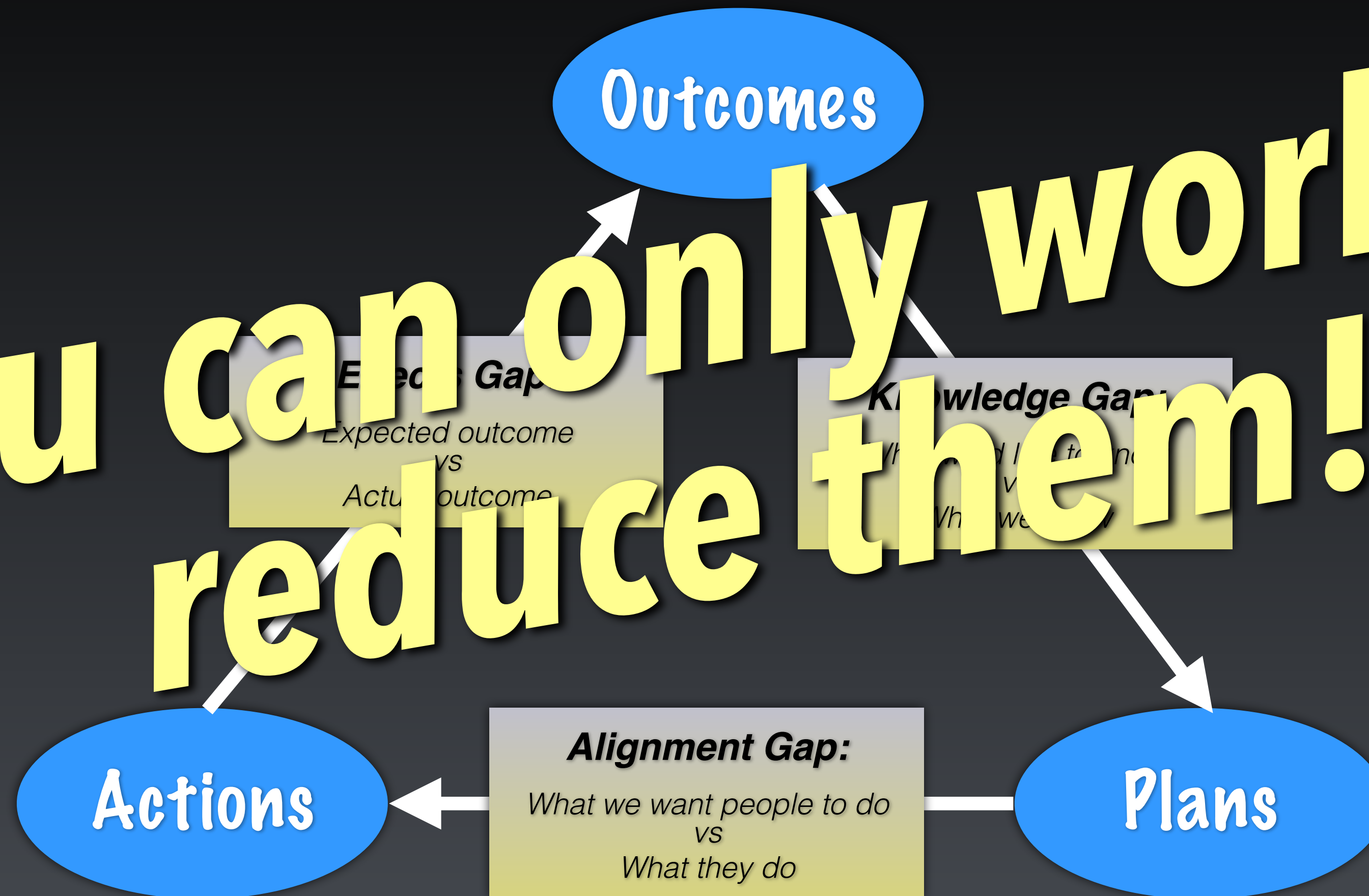


You can't close the gaps!

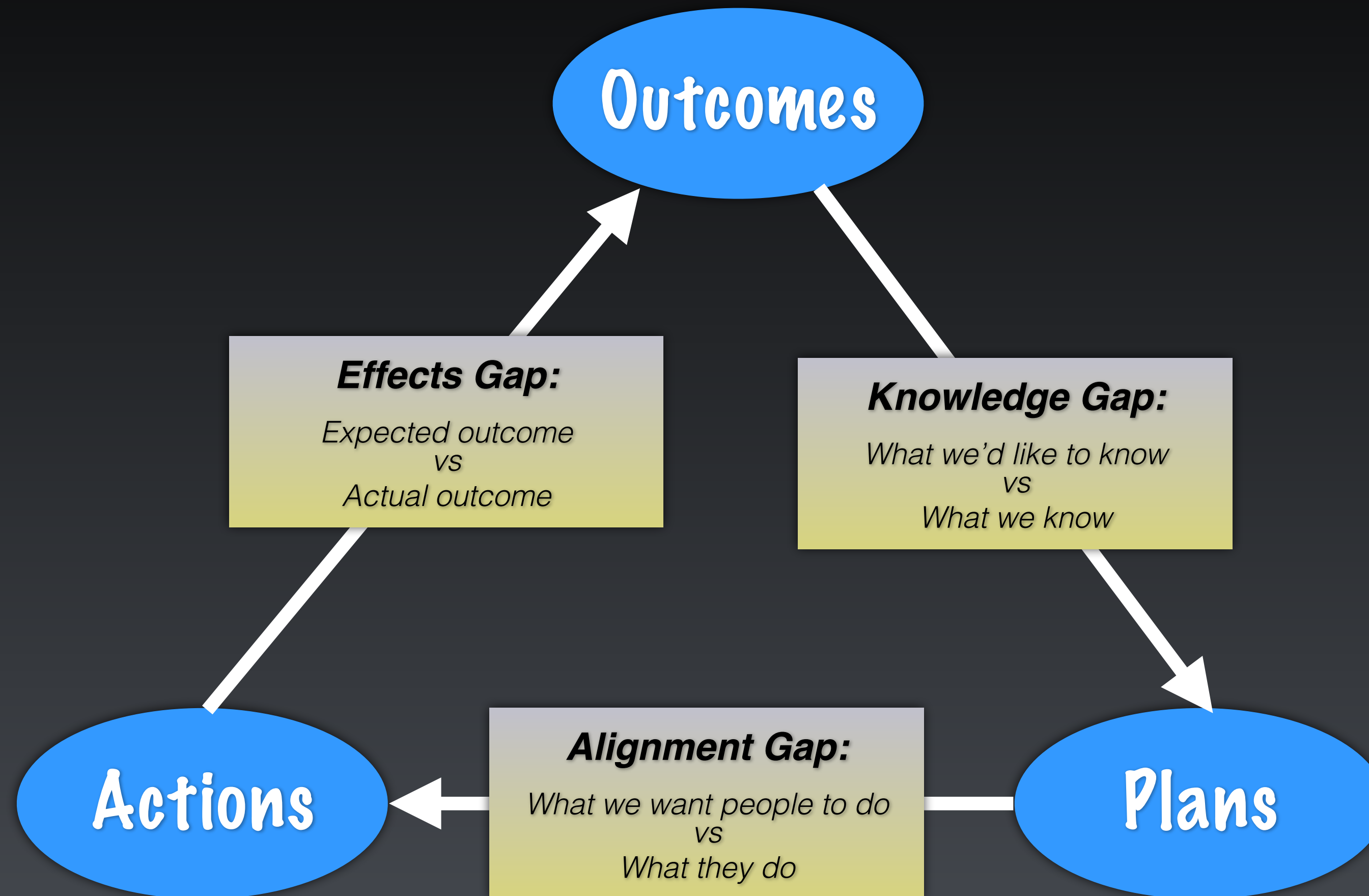




**You can only work to
reduce them!**



Best Way to Reduce Gaps - Speed Up!



Best Way to Reduce Gaps - Speed Up!

Work In Small Steps!

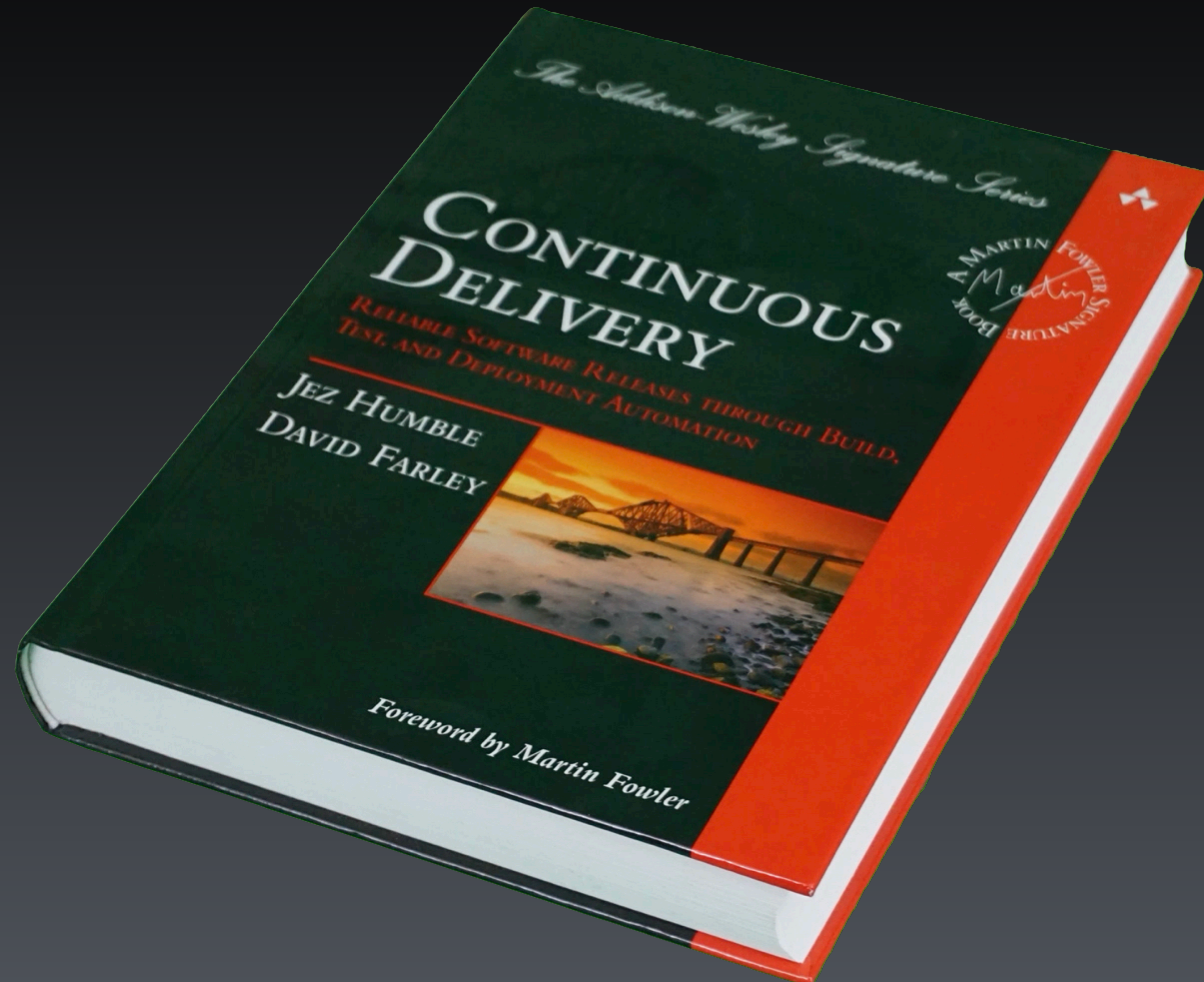
The diagram illustrates a cycle of four blue ovals: 'Outcome' at the top, 'Action' on the left, 'Knowledge' on the right, and 'Performance' at the bottom. Arrows connect them in a clockwise cycle. Three yellow boxes labeled 'Effects Gap', 'Knowledge Gap', and 'Performance Gap' are placed between the ovals, each with a definition. A diagonal line from 'Outcome' to 'Performance' is labeled 'New' and 'Original'.

Effects Gap:
Expected outcome vs
Actual outcome

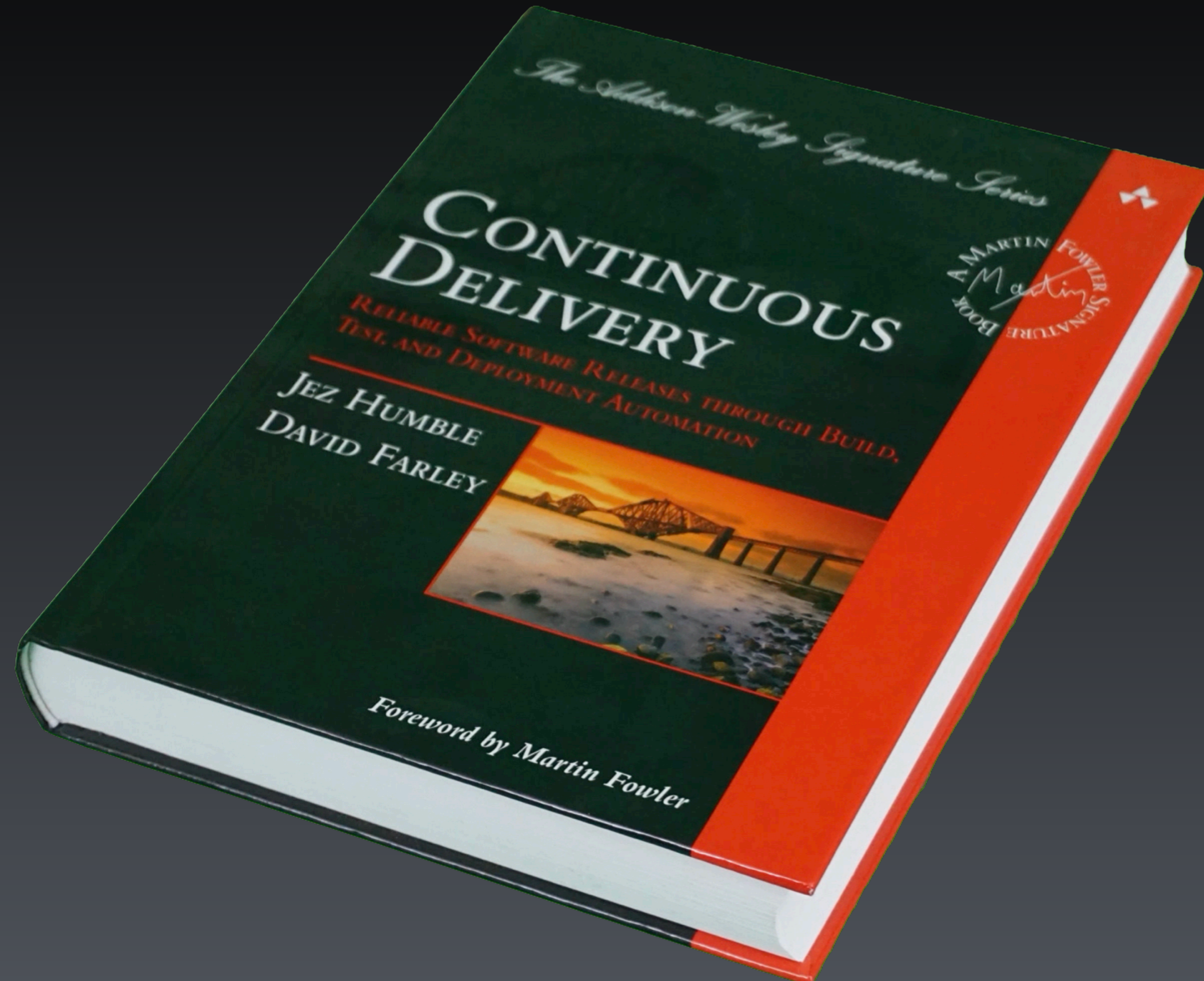
Knowledge Gap:
What we know vs
What we need to know

Performance Gap:
What we want people to do vs
What they do

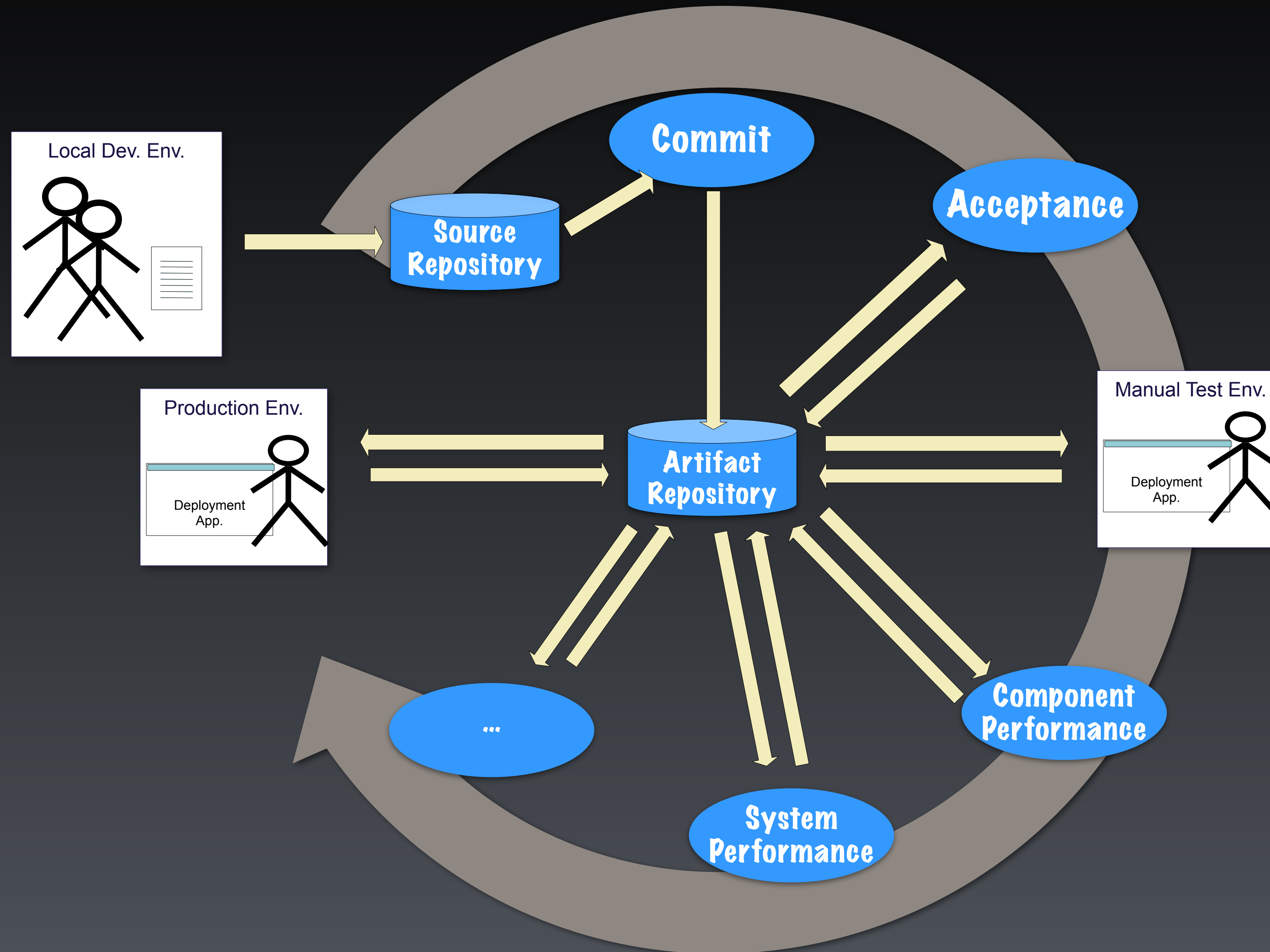
**Work In
Small Steps!**



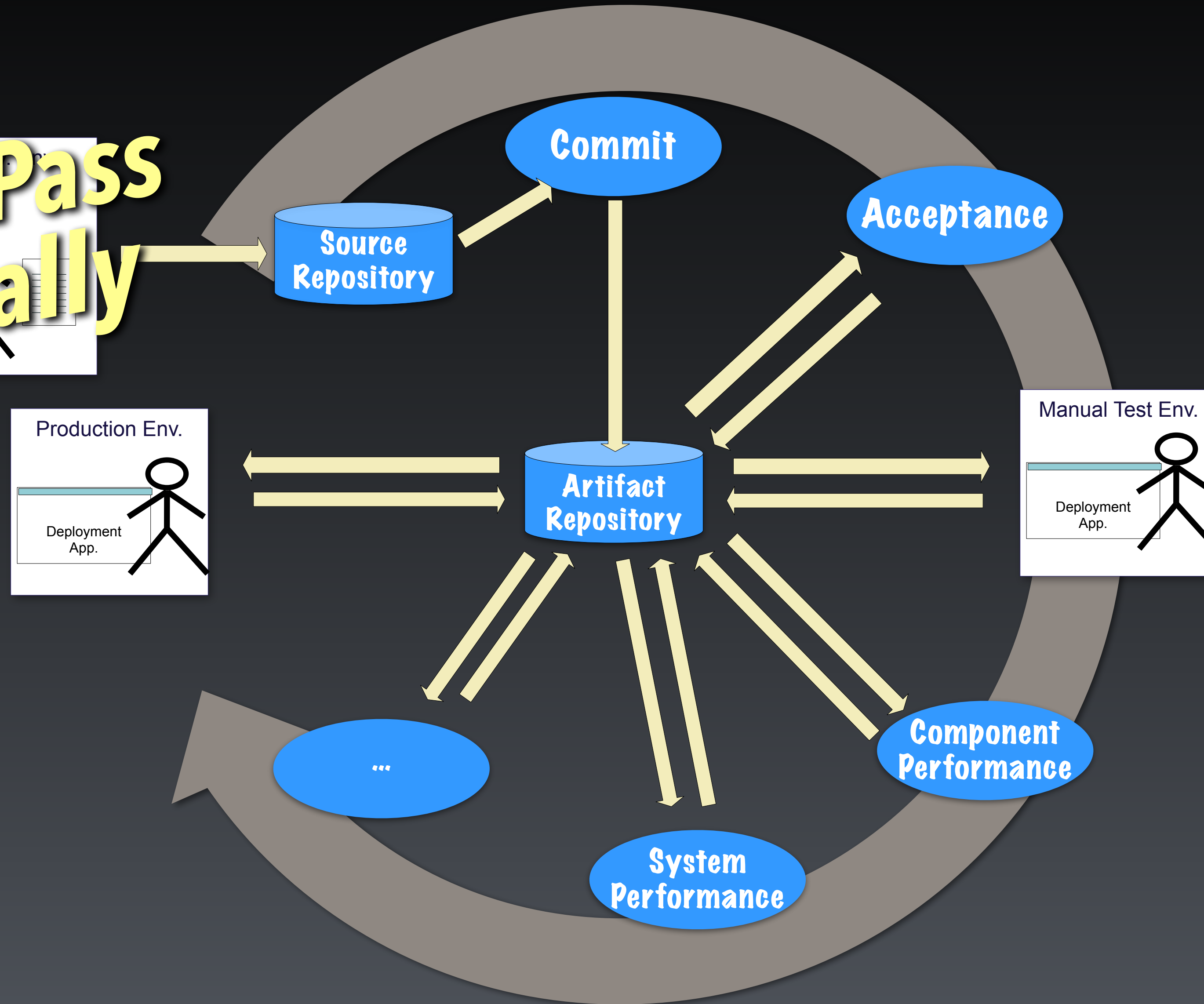
**"Working so our
Software is ALWAYS in a
Releasable State"**



So What Determines
'Releasability'?

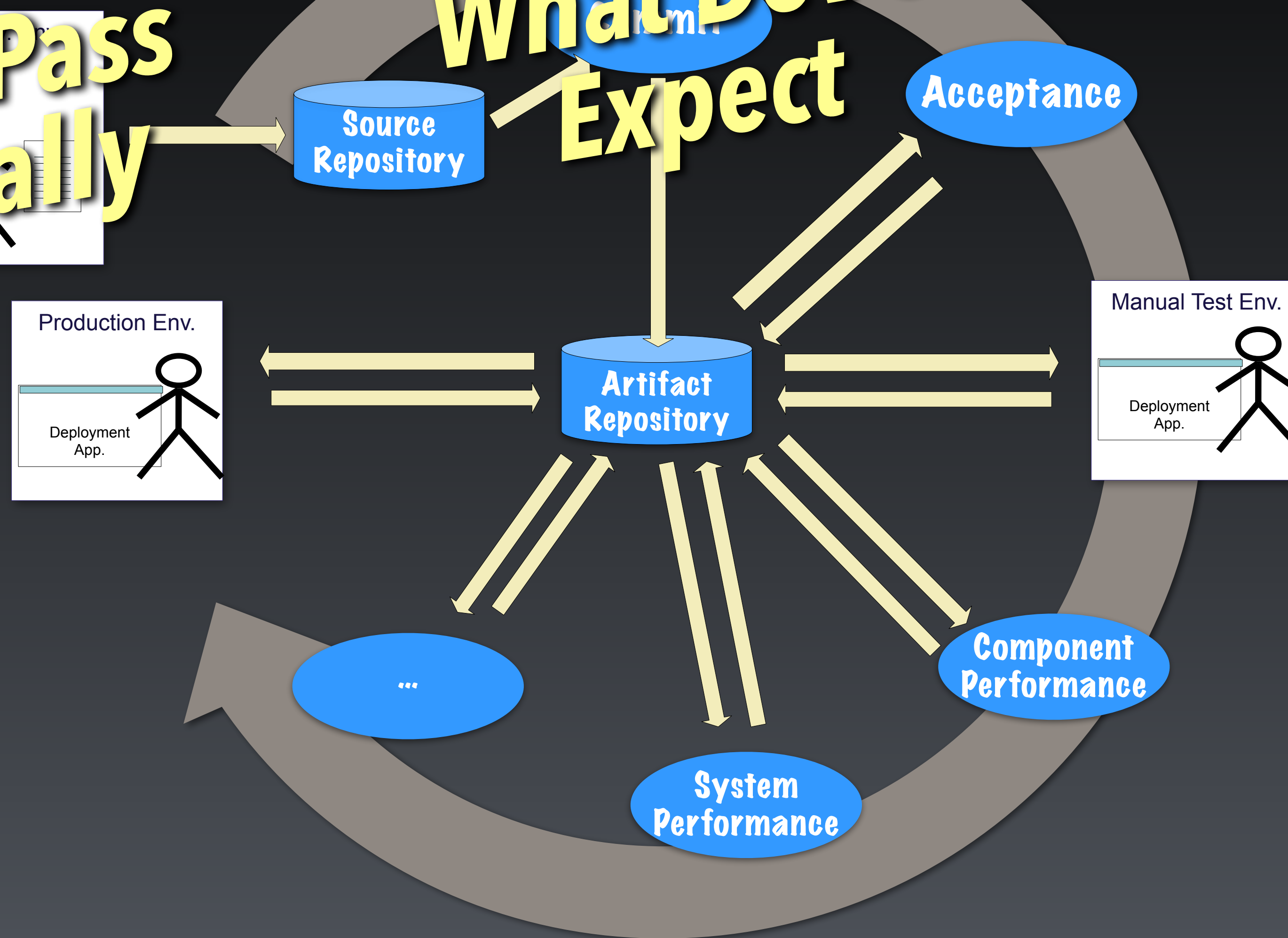


Test Pass Locally



Test Pass Locally

Code Does What Dev's Expect



Test Pass Locally

Code Does What Dev's Expect

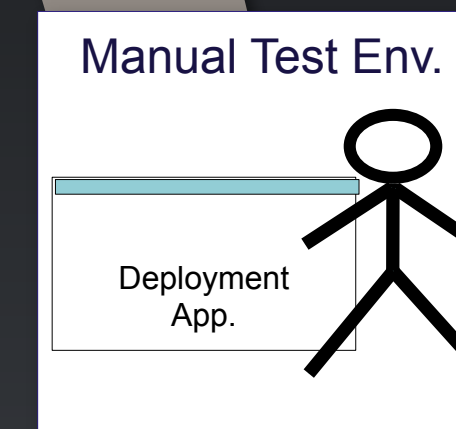
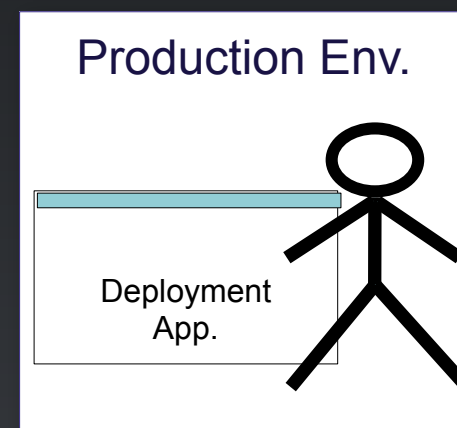
Code Does What Users Want



Source Repository

Artifact Repository

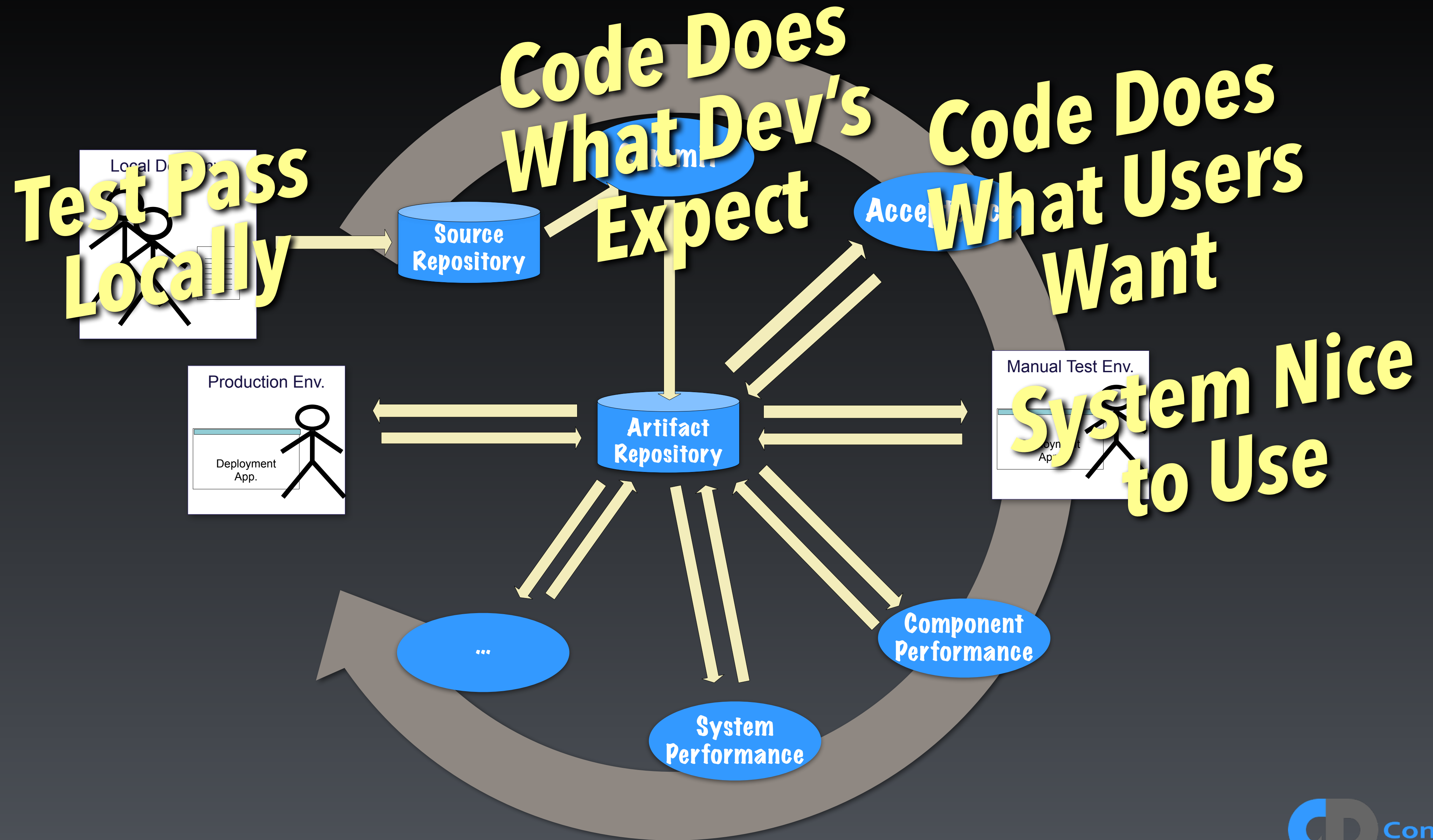
Acceptance

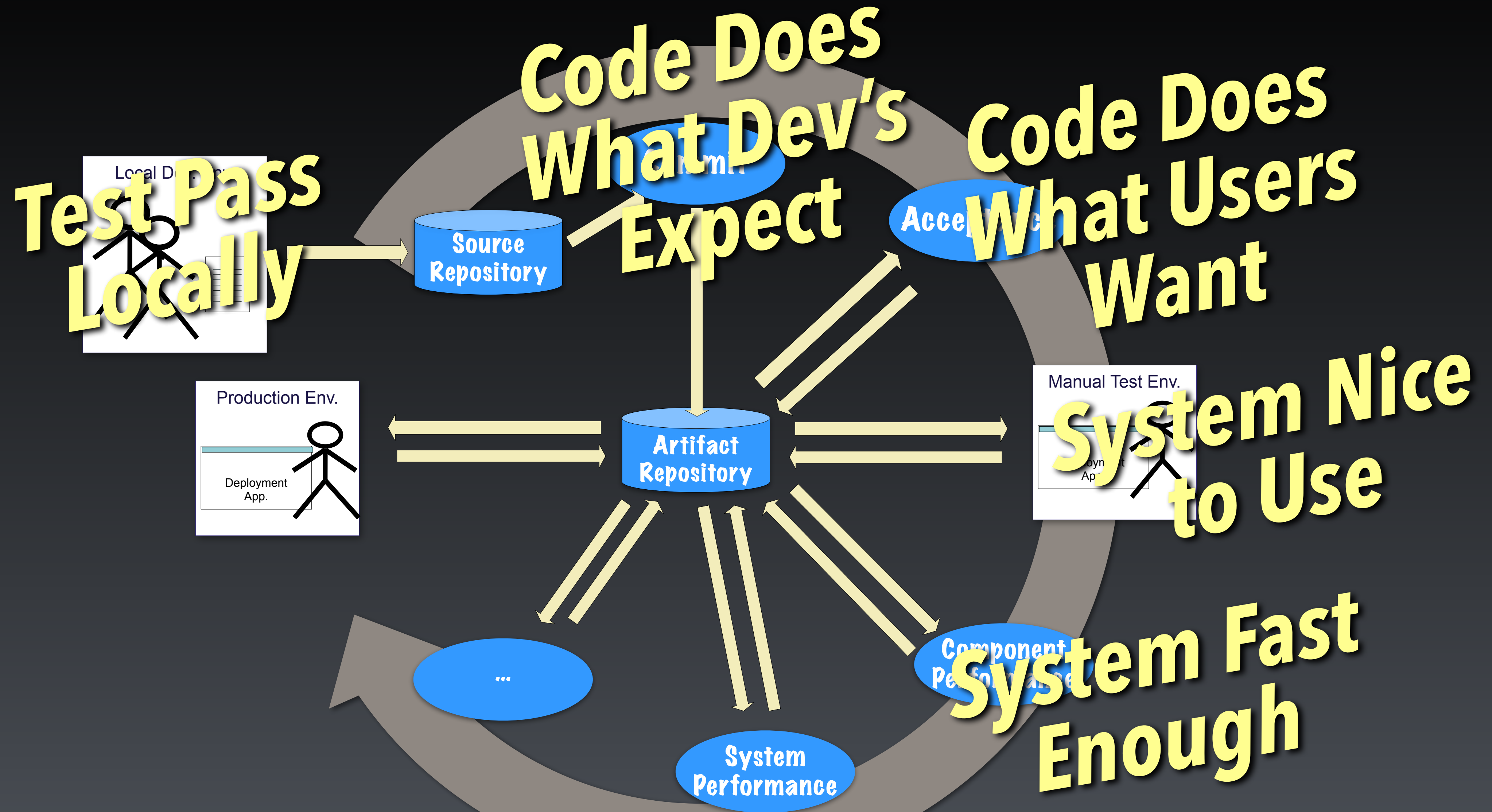


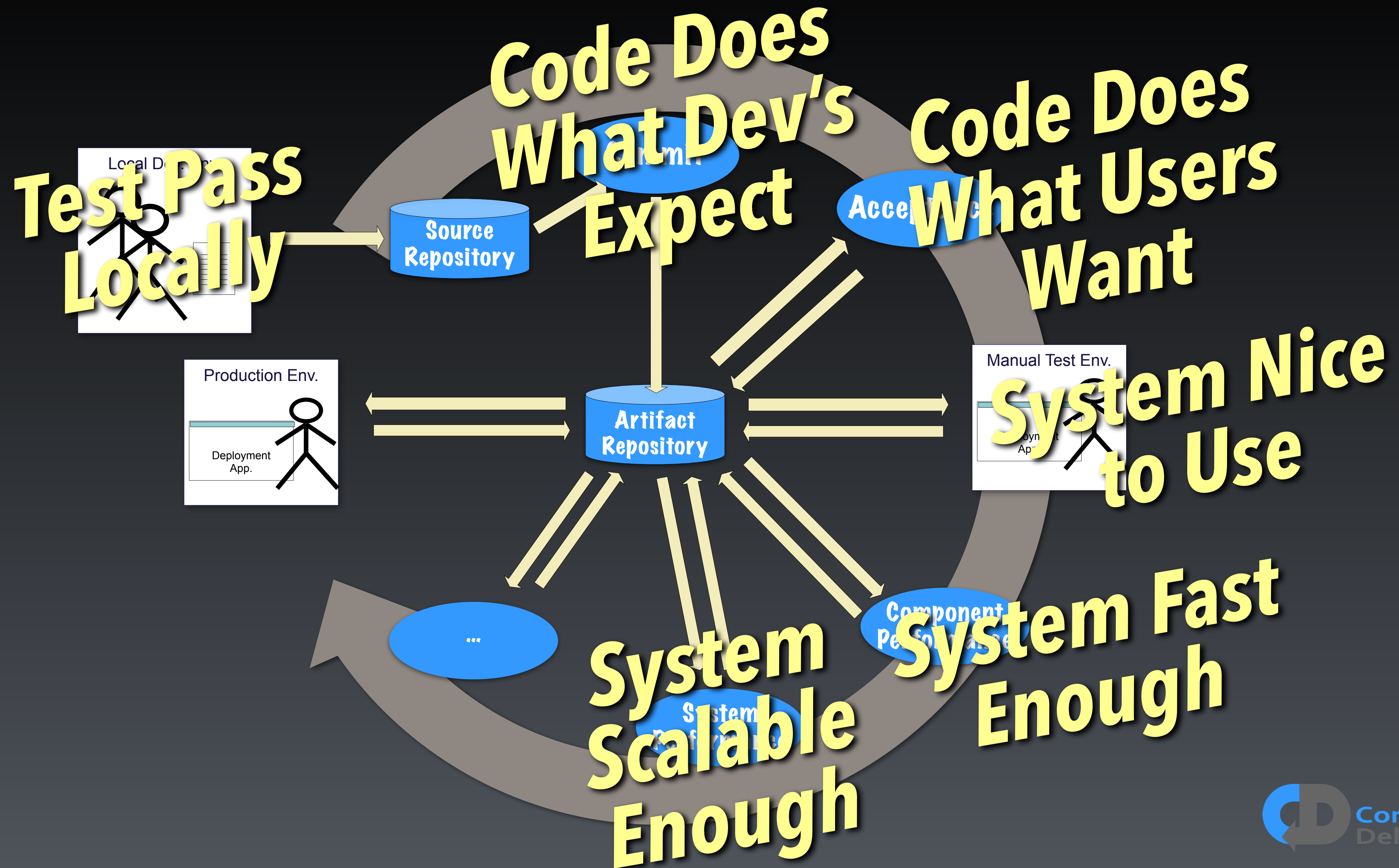
...

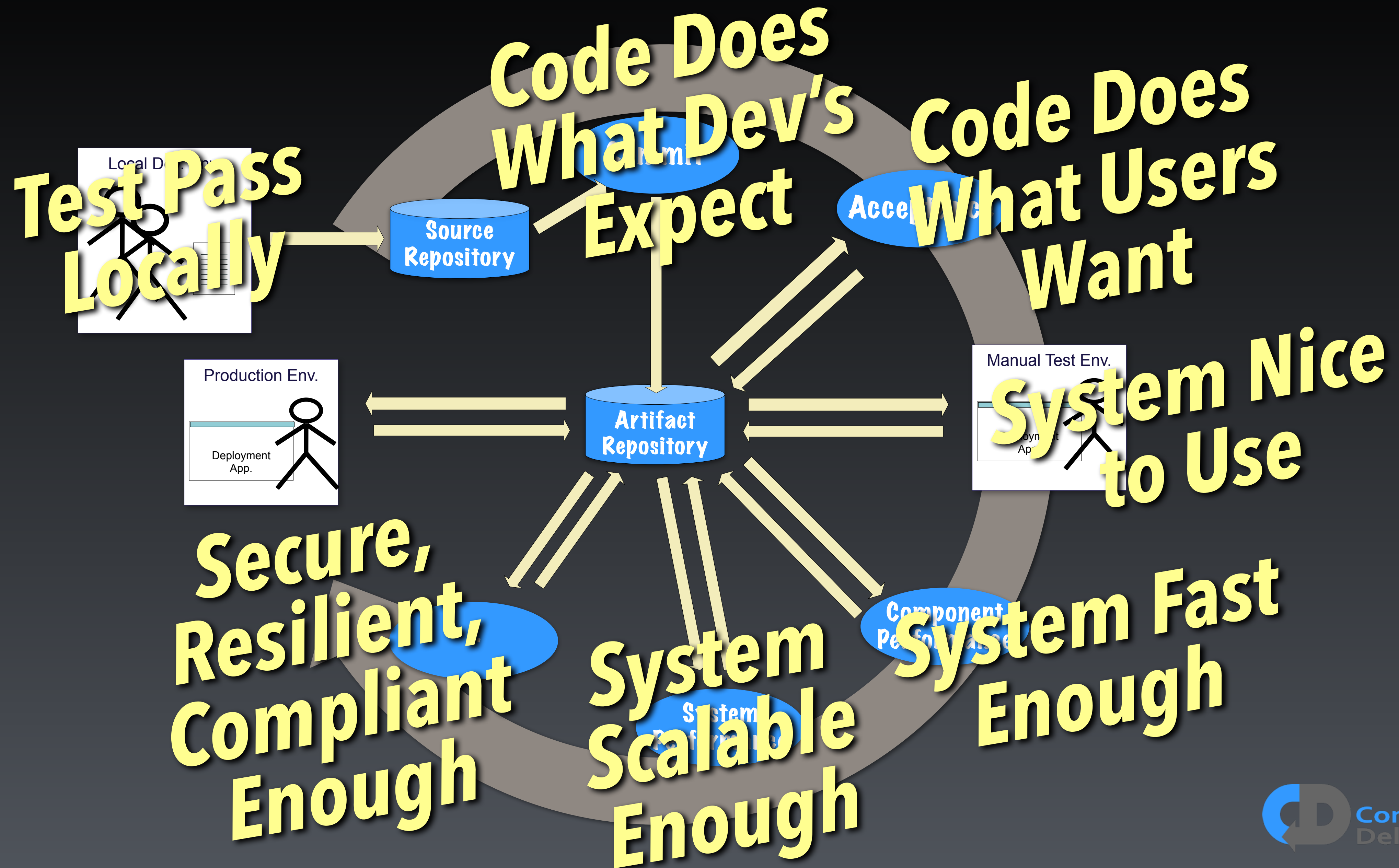
System Performance

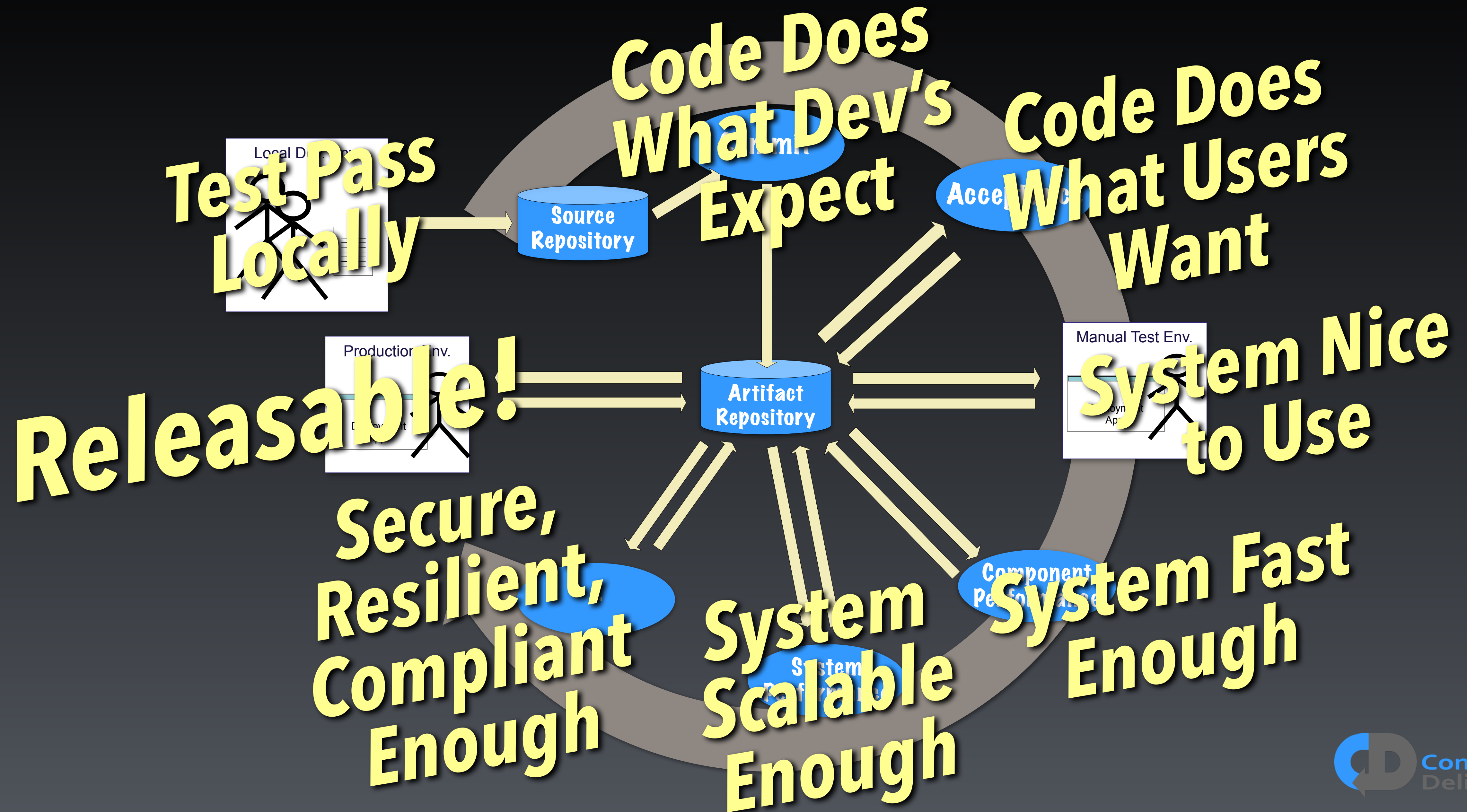
Component Performance

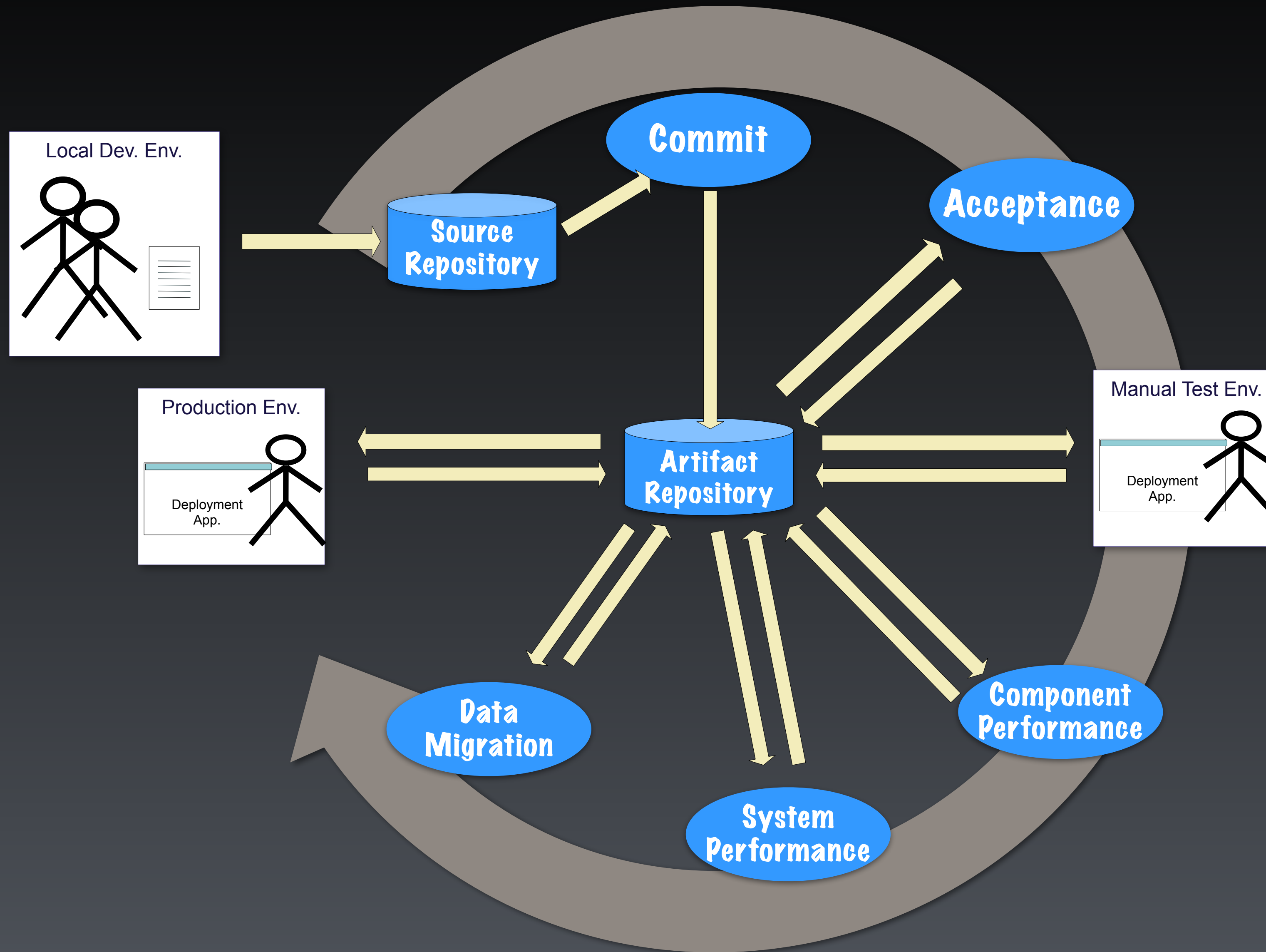




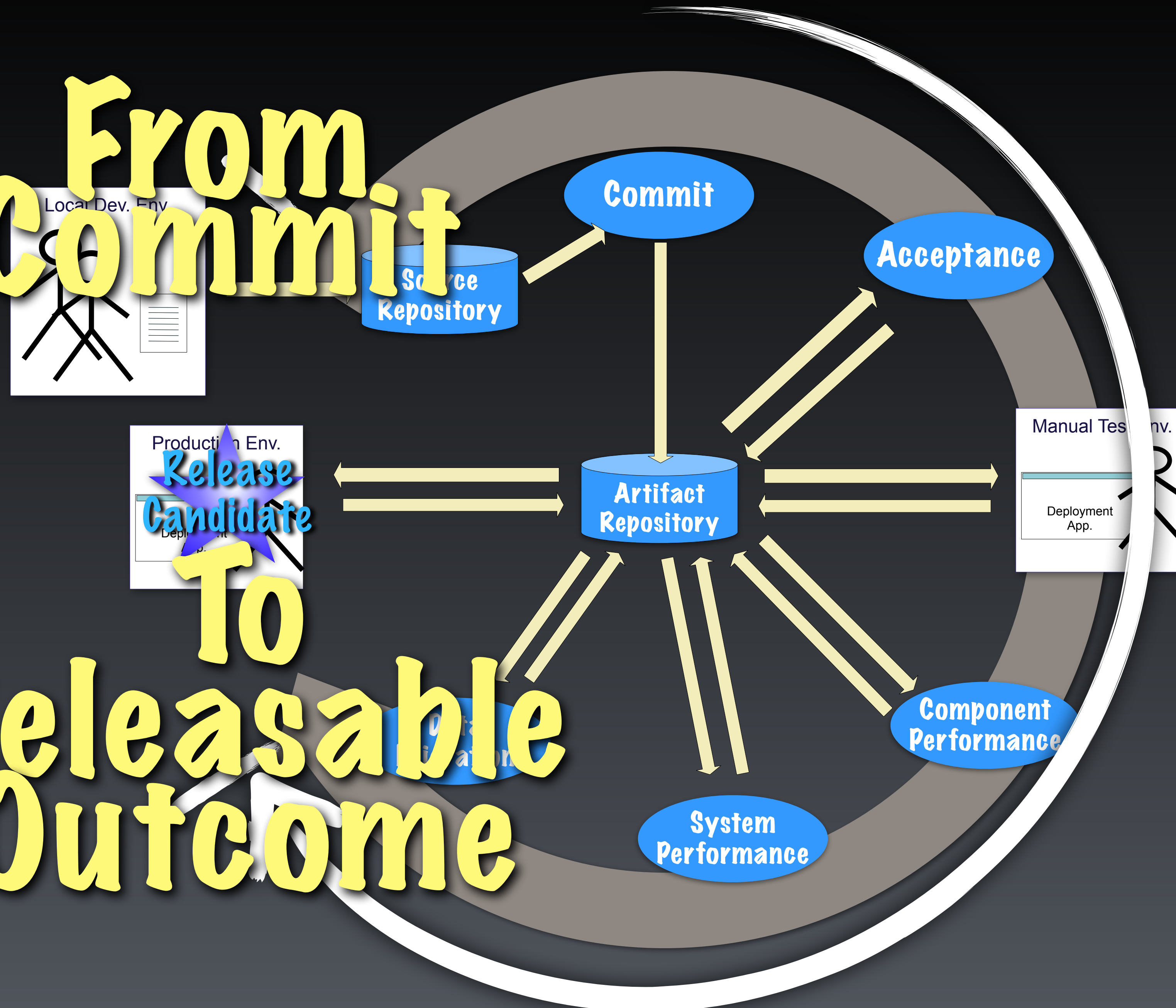








From Commit To Releasable Outcome



Confident to Make Progress

- We Gain Confidence by Testing Our System Thoroughly
- Testing Thoroughly, Means Automated Testing
- I Think of this as ***Engineering*** for Software

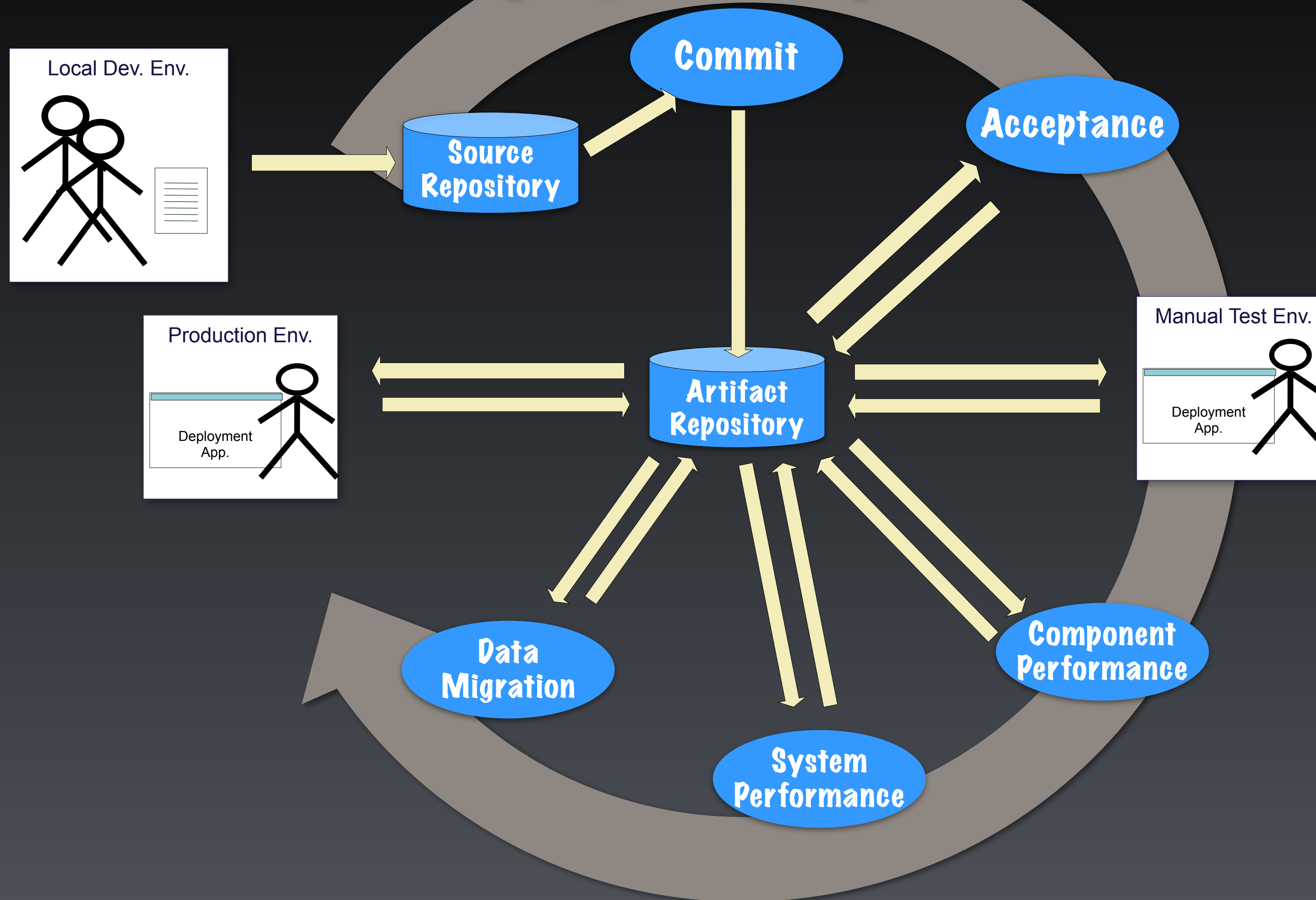
Confident to Make Progress

- We Gain Confidence by Testing Our System Thoroughly
- Testing Thoroughly, Means Automated Testing
- I Think of this as ***Engineering*** for Software

**True
Software Engineering**

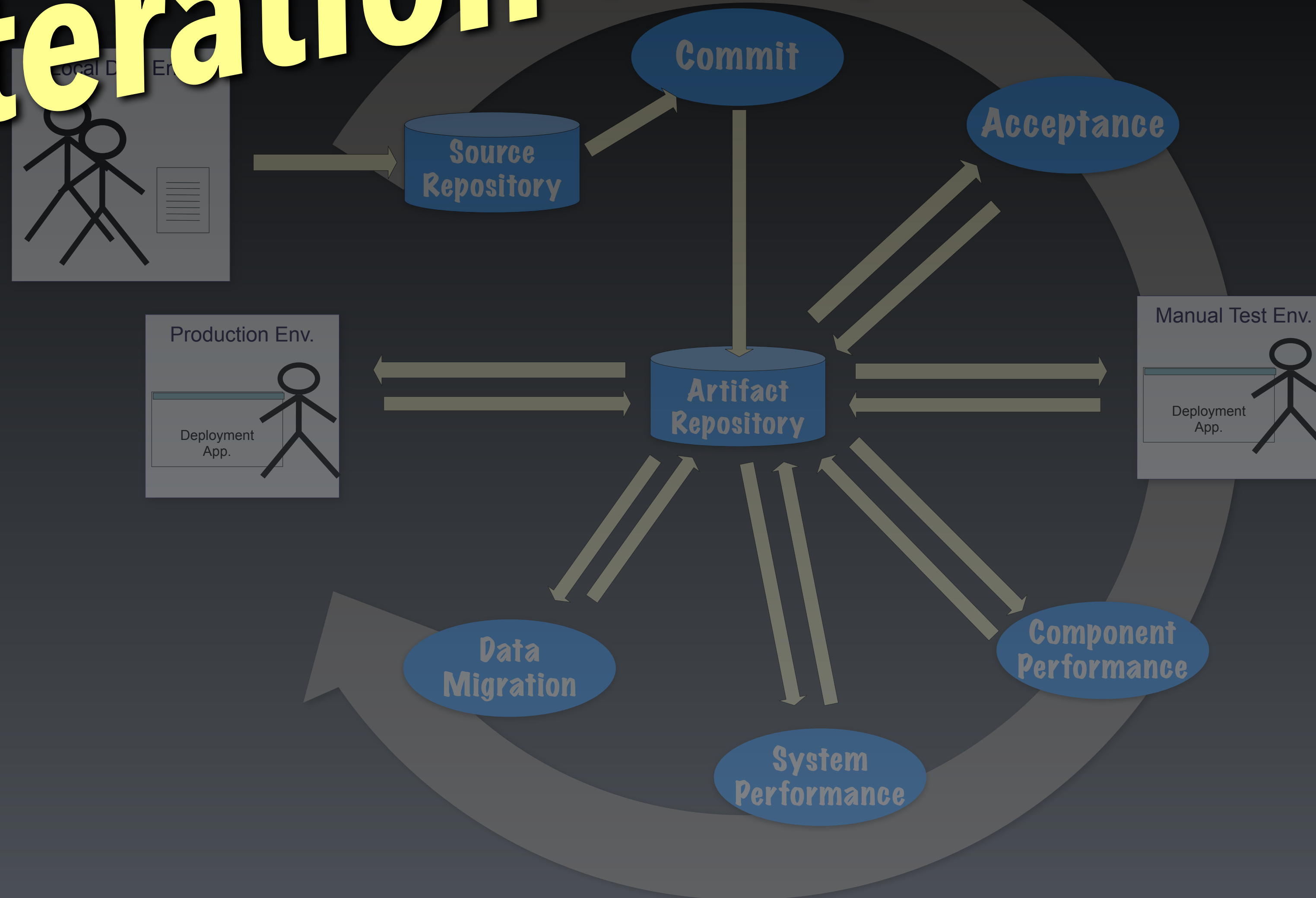
Optimise for Learning

Deployment Pipeline



Optimise for Learning

Iteration Deployment Pipeline

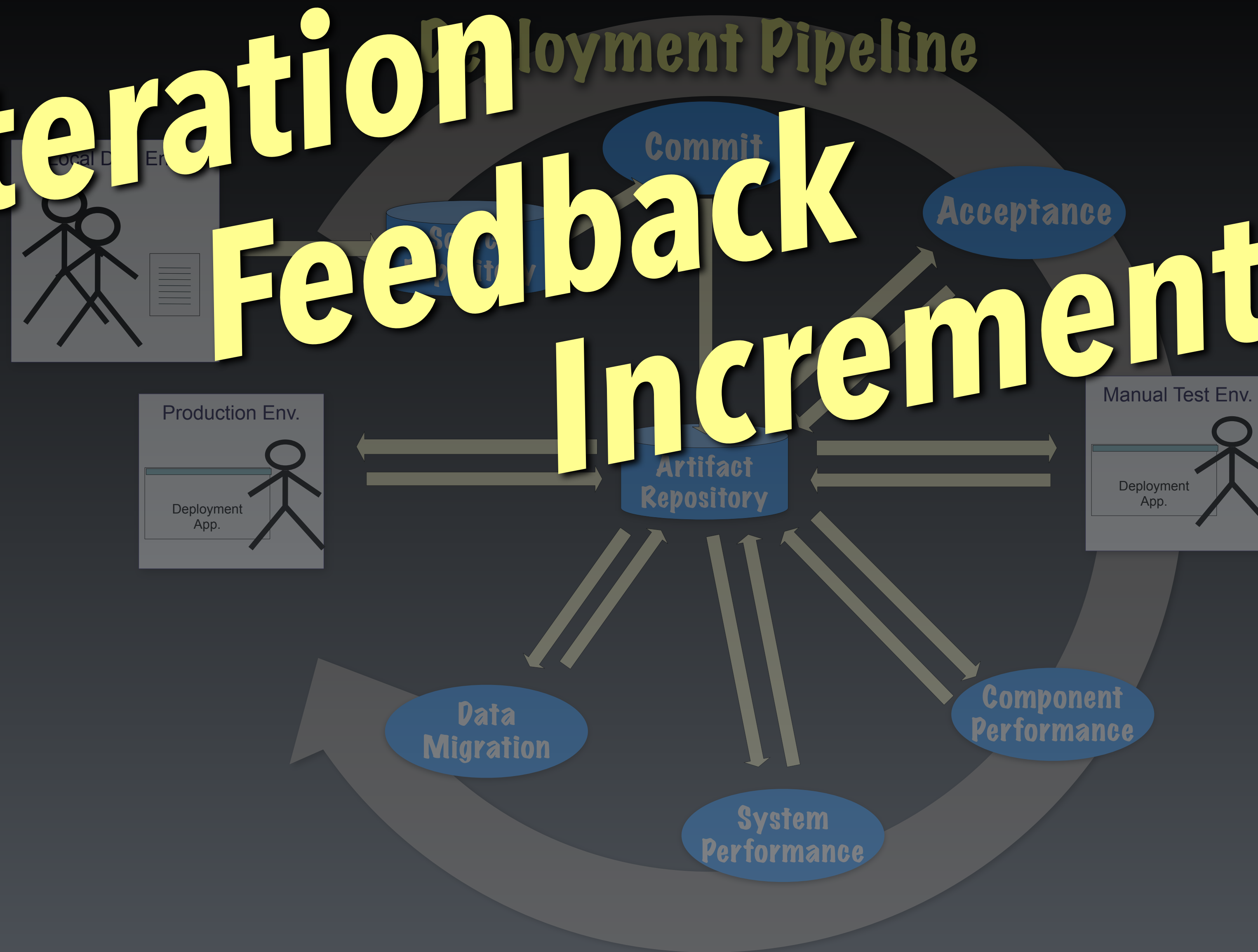


The diagram shows a continuous cycle of development and deployment. It starts with 'Local Development' (represented by two stick figures), followed by 'Staging Environment' (represented by a document icon), then 'Commit' (in a blue oval), 'Deploy' (in a blue oval), and 'Acceptance' (in a blue oval). A large blue arrow points from 'Acceptance' back to 'Local Development', completing the loop. The text 'Iteration Feedback' is written in large, bold, yellow letters across the center of the diagram.



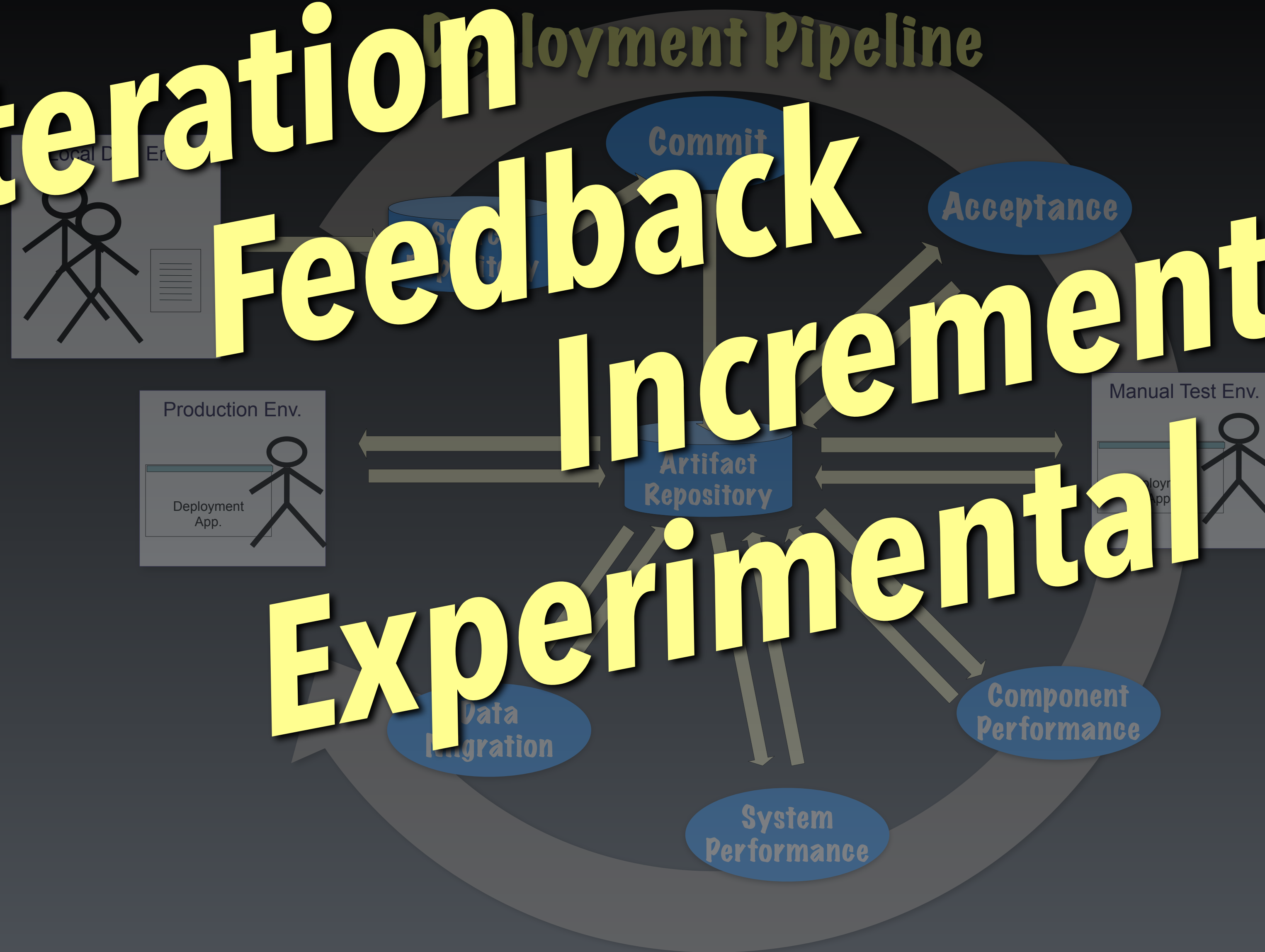
Optimise for Learning

Iteration Feedback Incremental



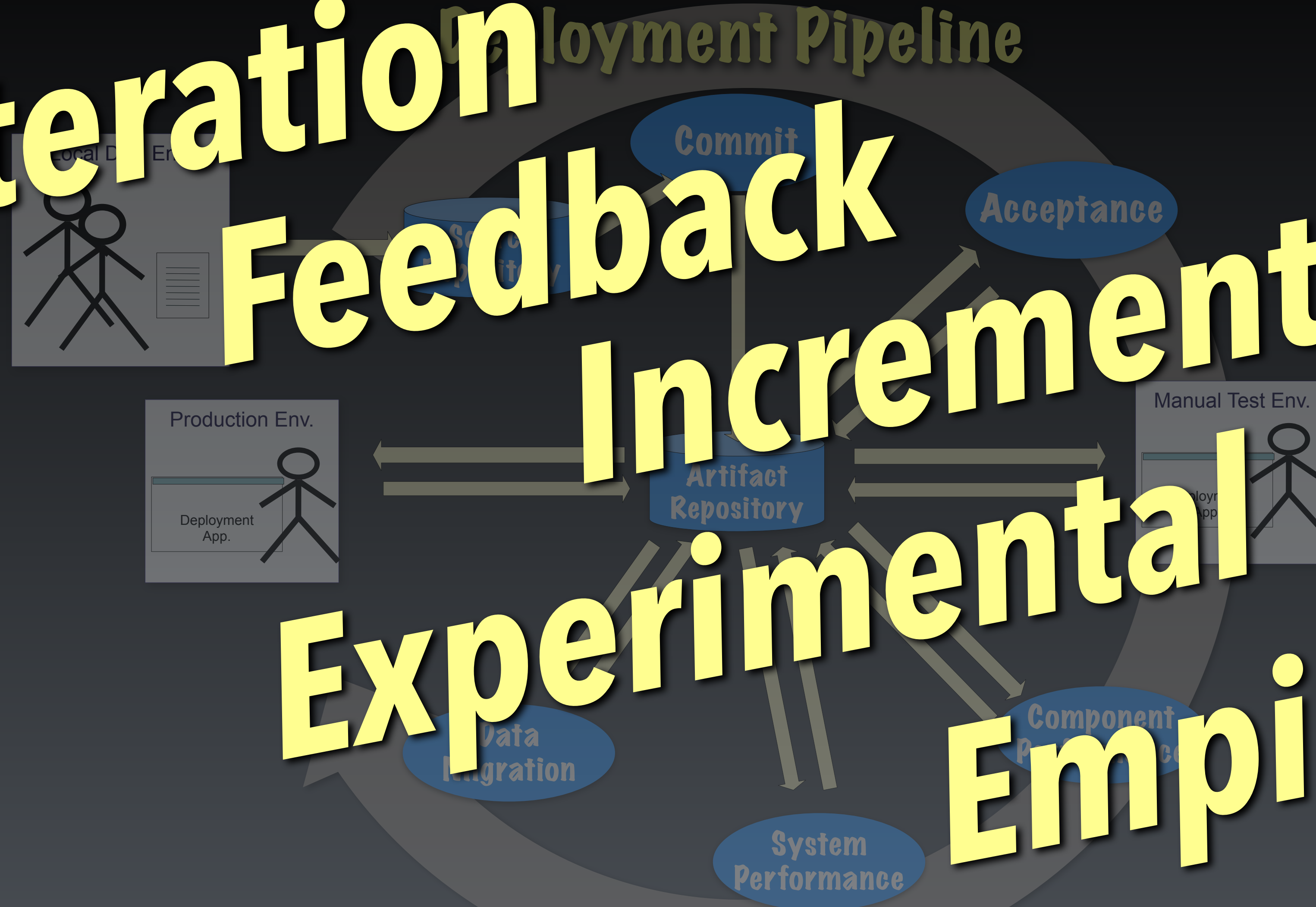
Optimise for Learning

Iteration Feedback Incremental Experimental



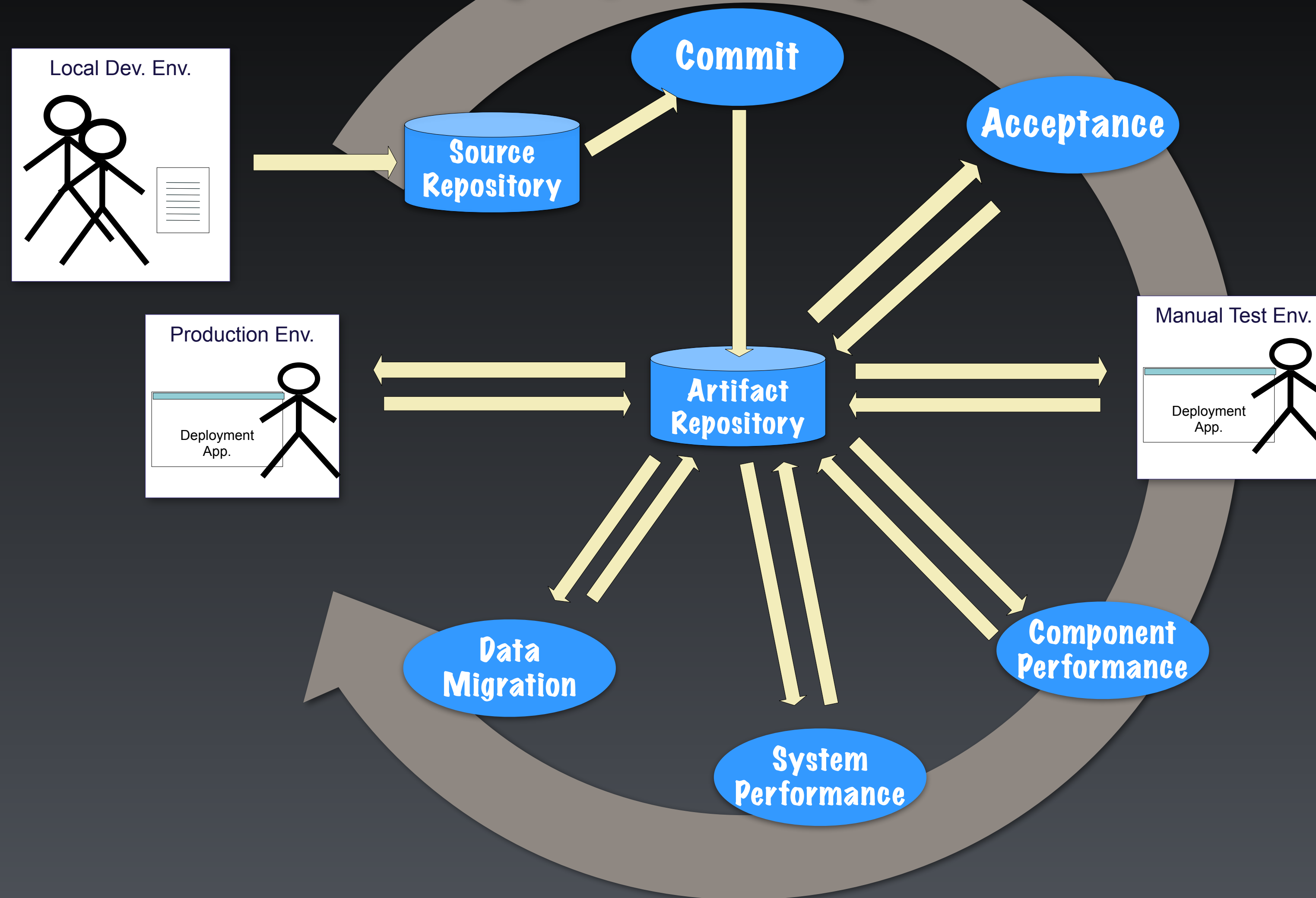
Optimise for Learning

Iteration
Feedback
Incremental
Experimental
Empirical



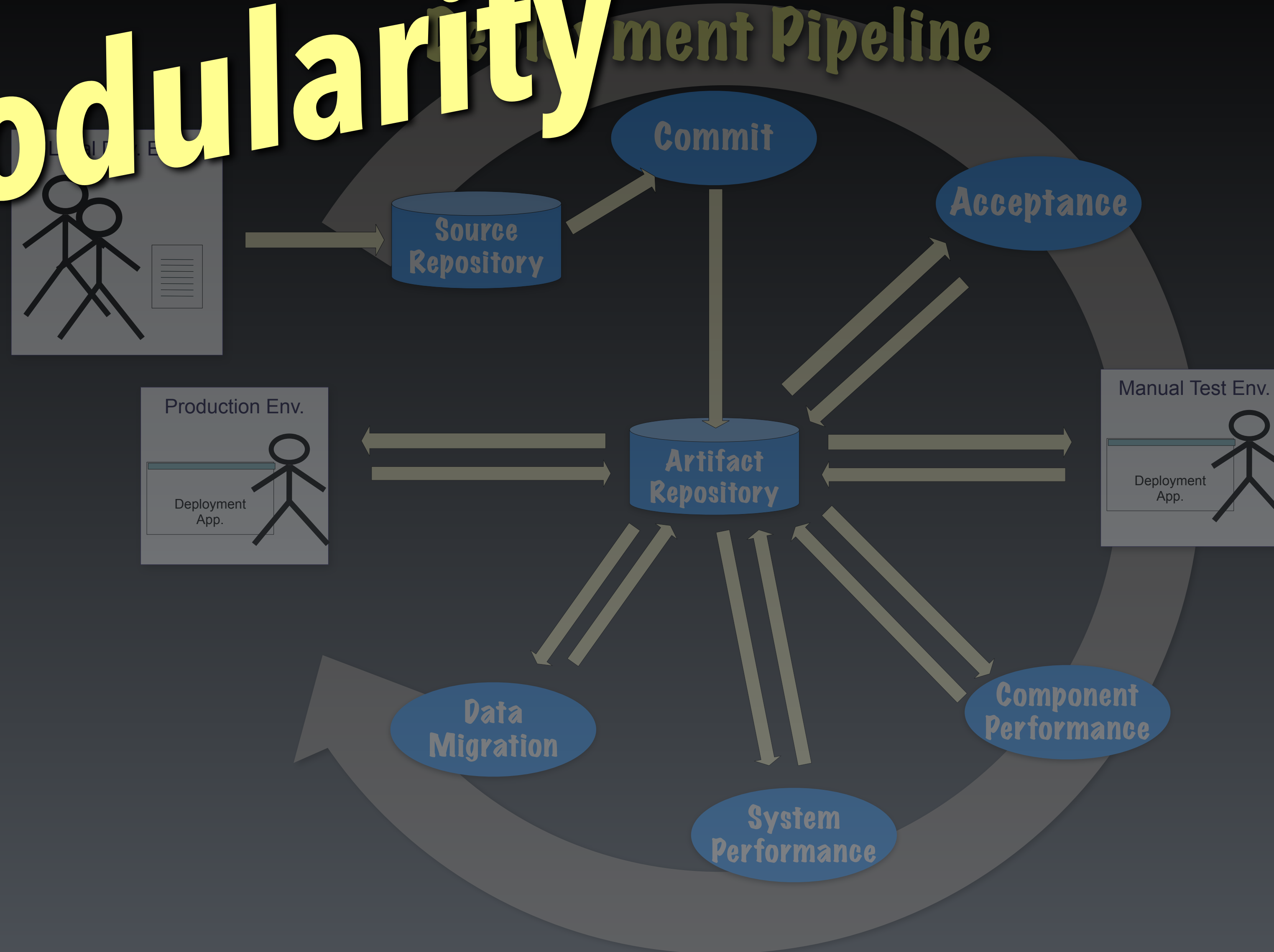
Managing Complexity

Deployment Pipeline



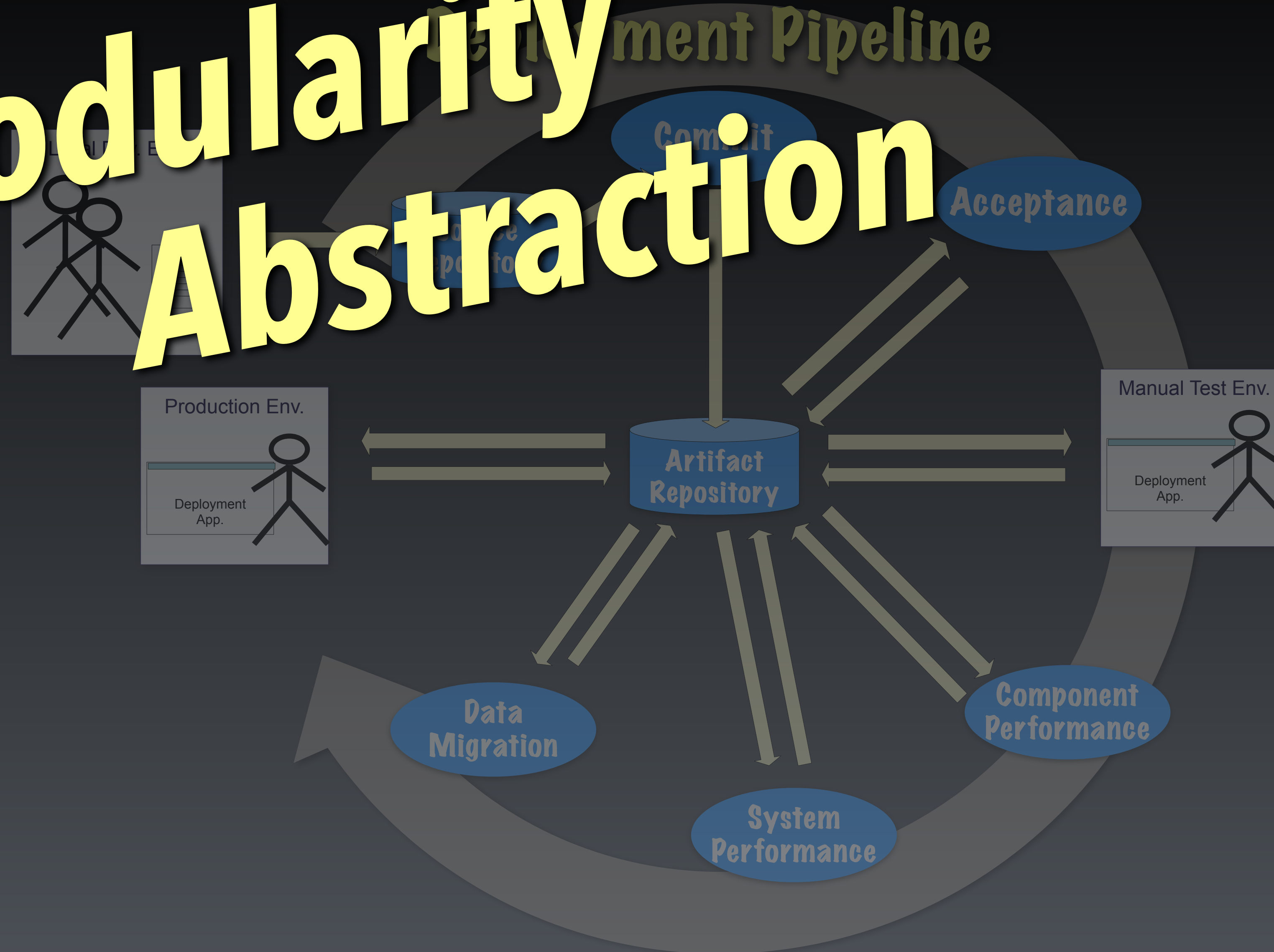
Managing Complexity

Modularity



Managing Complexity

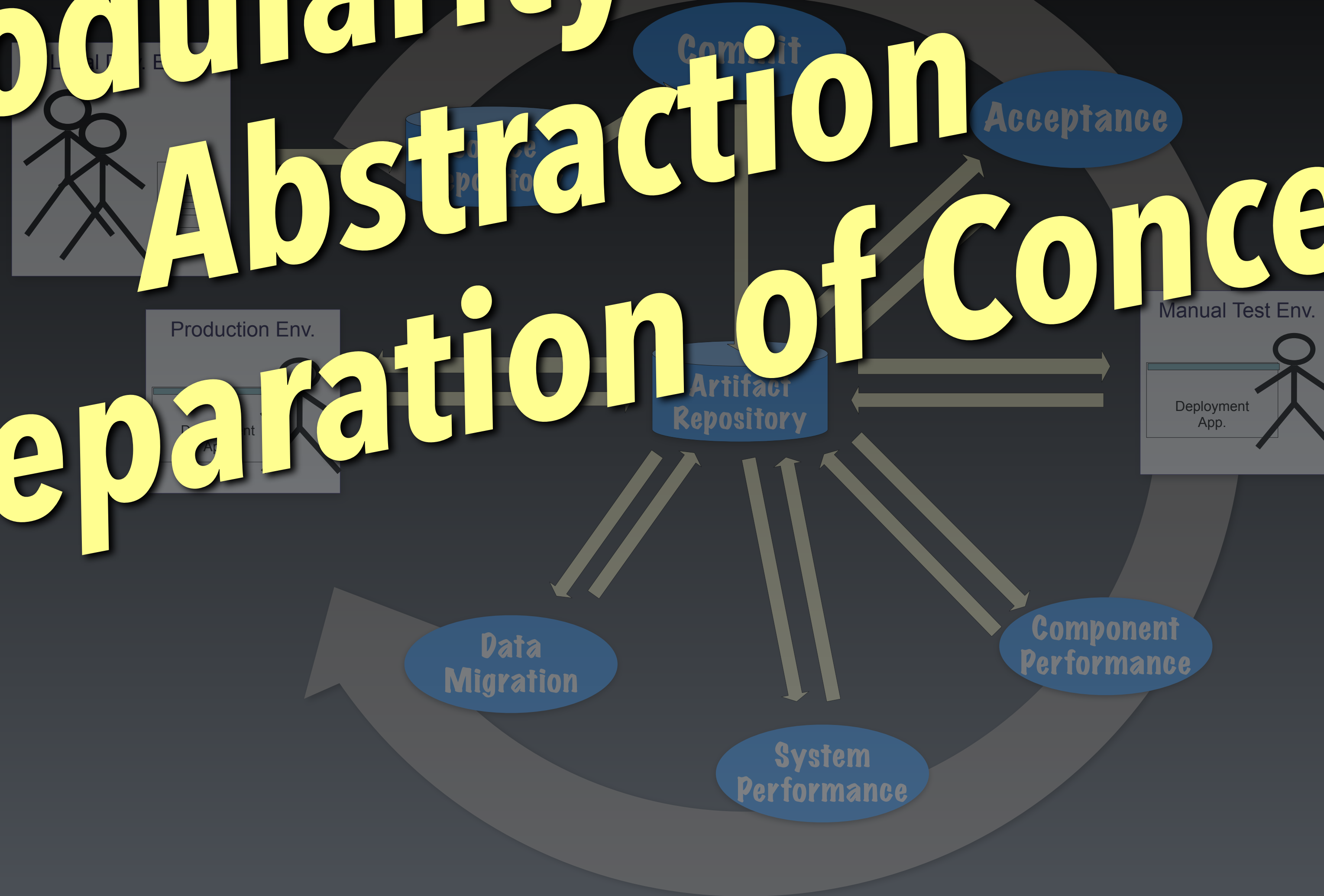
Modularity Abstraction



Managing Complexity

Deployment Pipeline

Modularity Abstraction Separation of Concerns



Managing Complexity

Deployment Pipeline

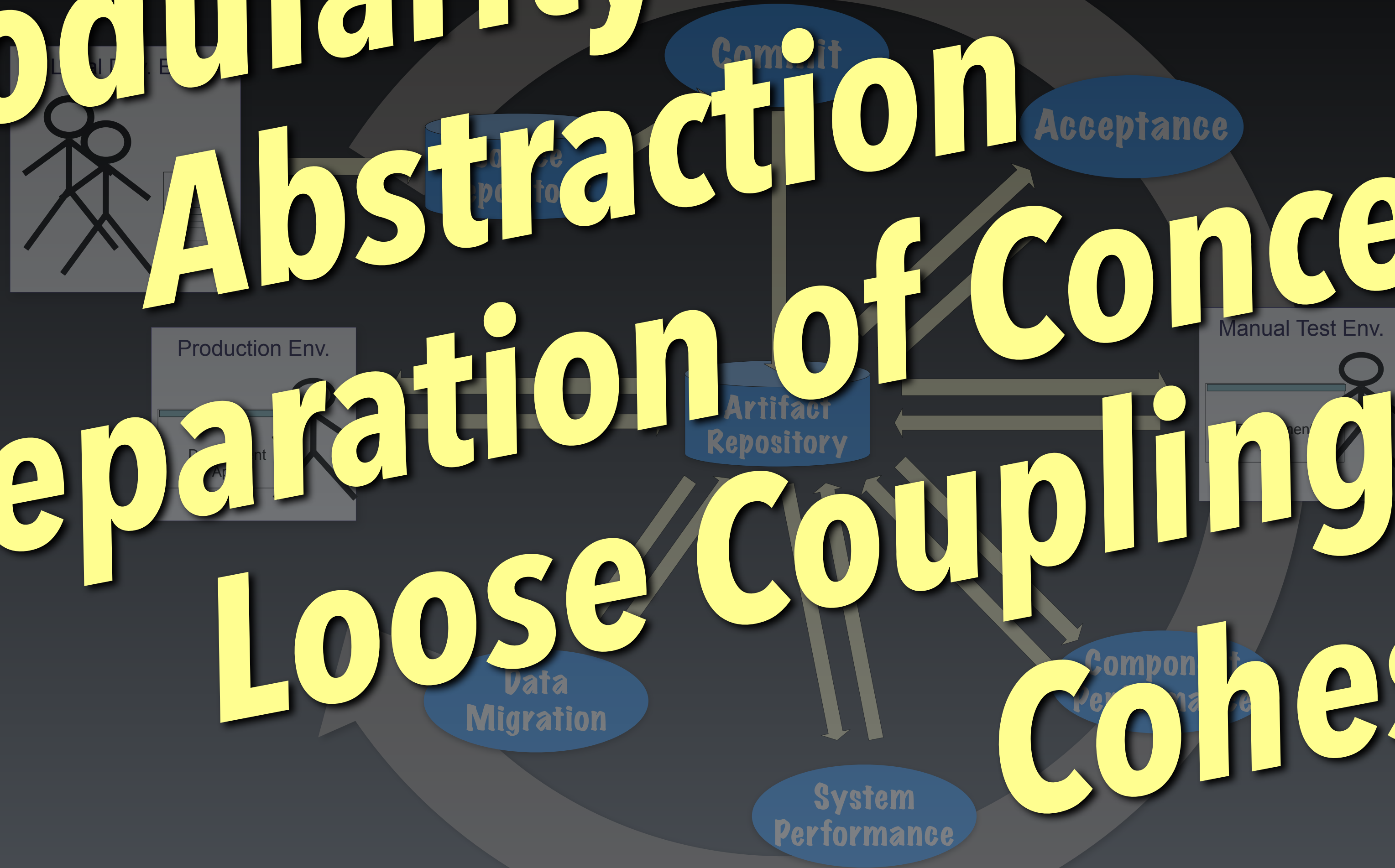
Modularity
Abstraction
Separation of Concerns
Loose Coupling

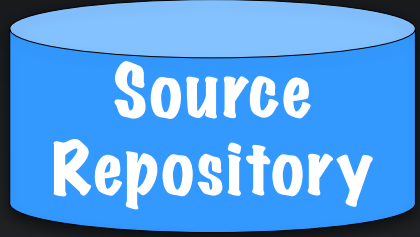
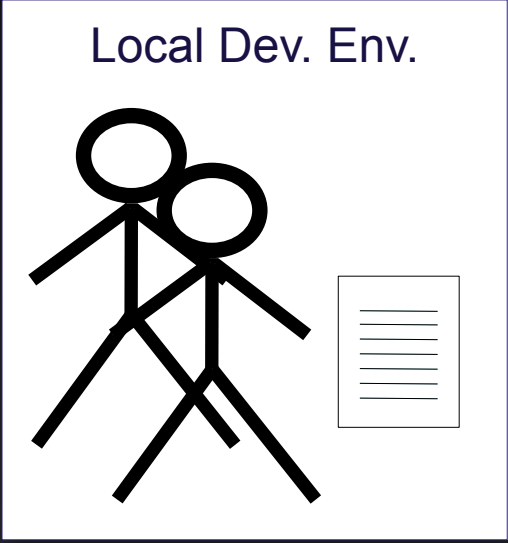


Managing Complexity

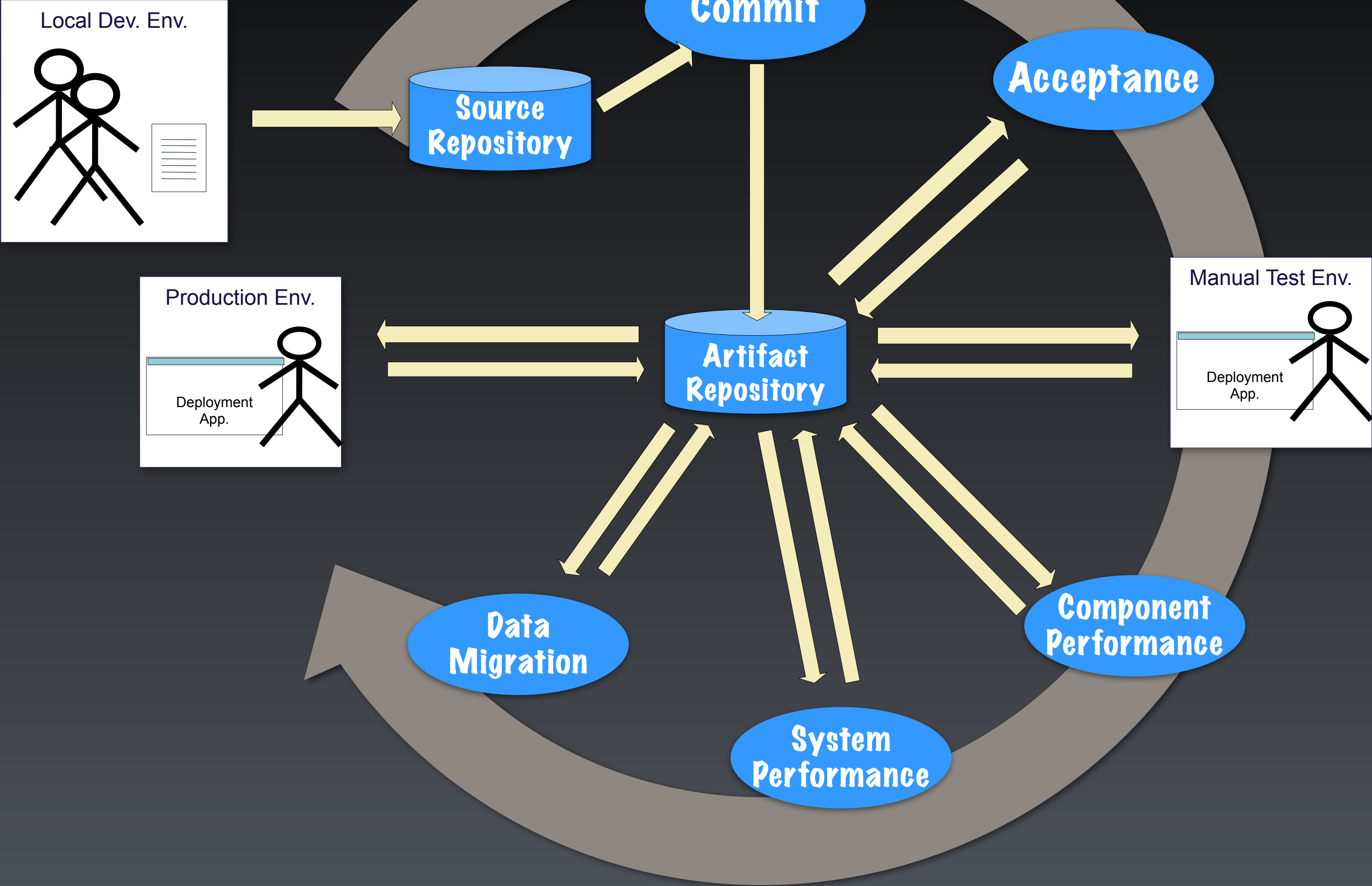
Deployment Pipeline

Modularity
Abstraction
Separation of Concerns
Loose Coupling
Cohesion

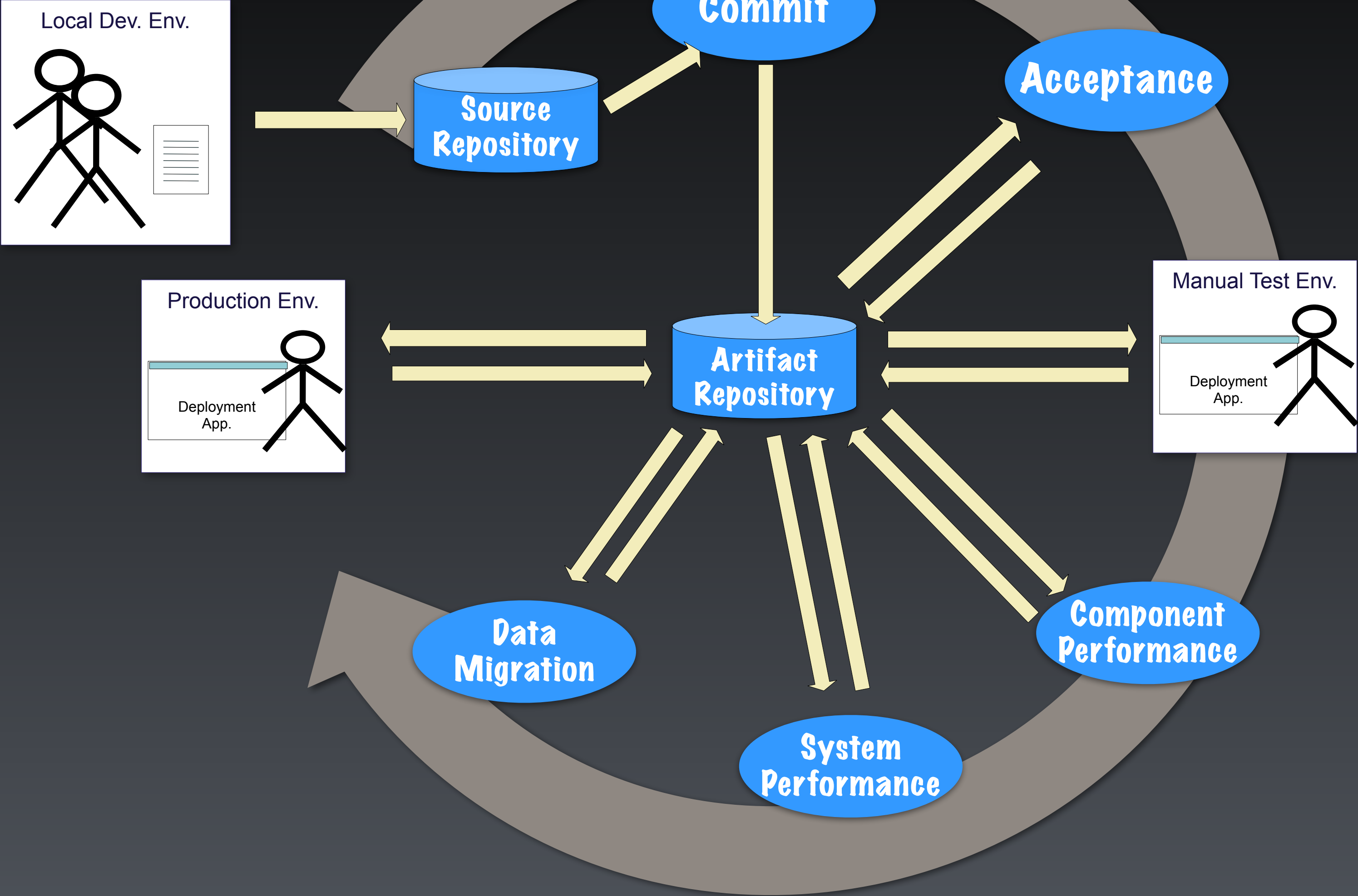




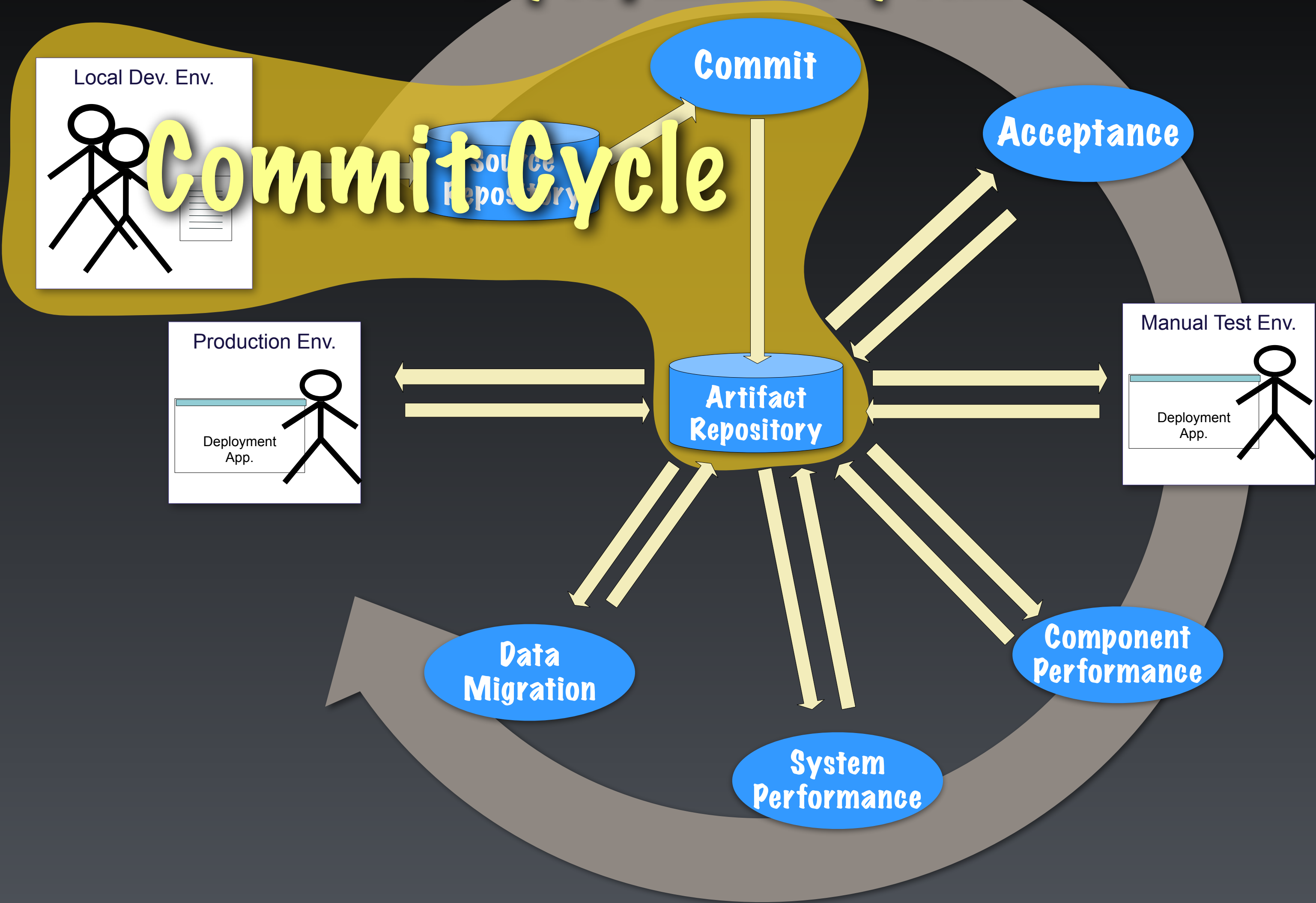
Deployment Pipeline



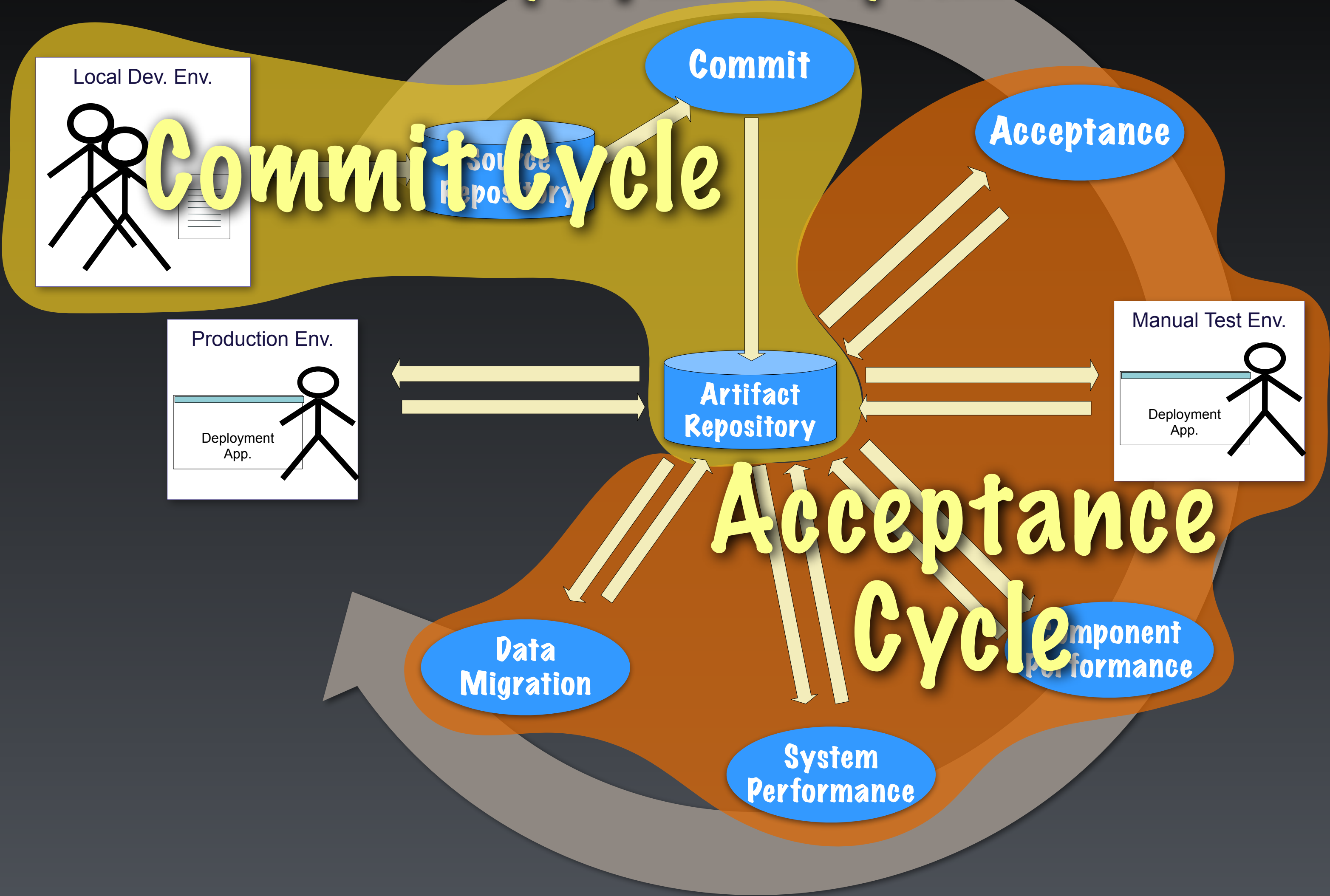
Deployment Pipeline



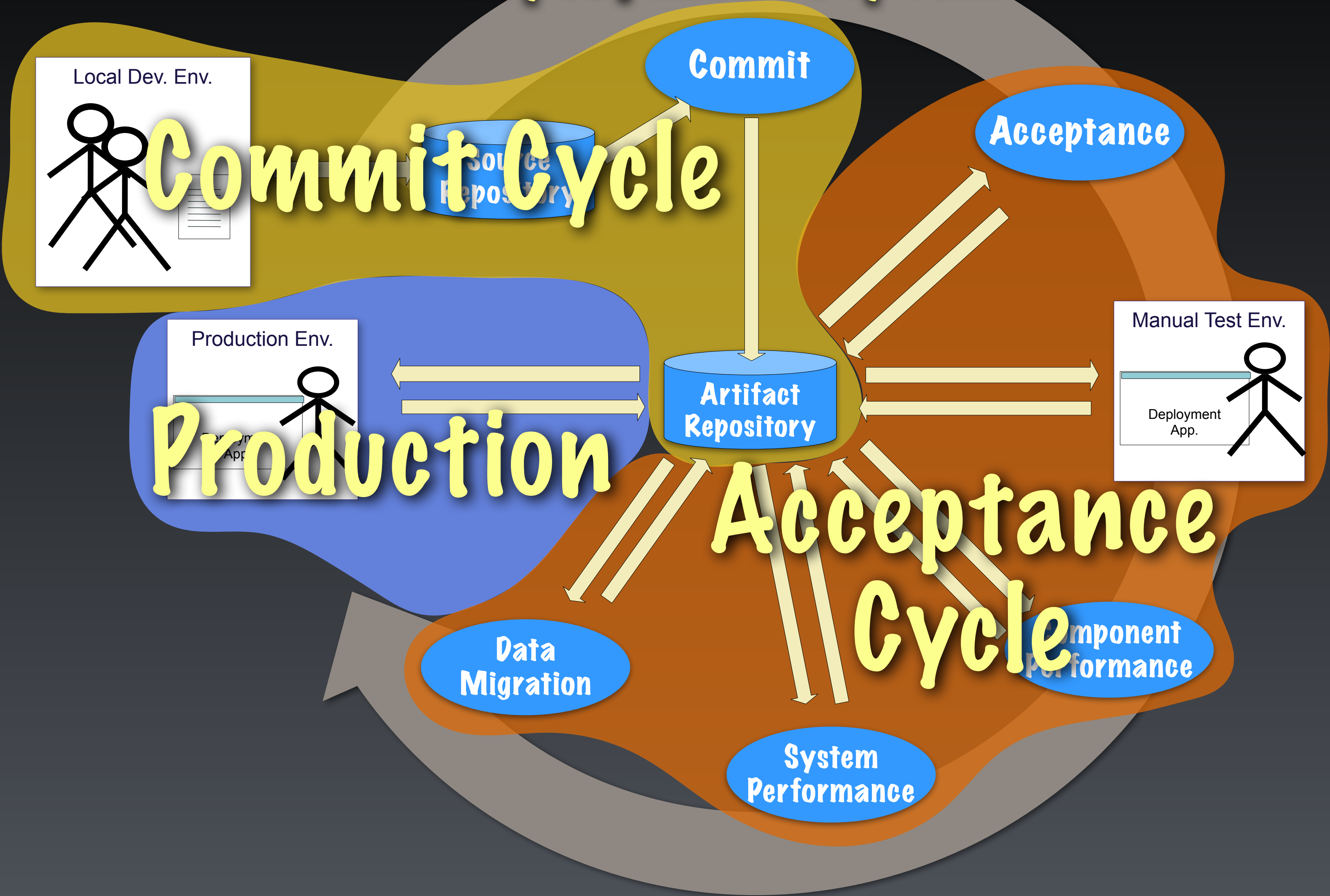
Deployment Pipeline



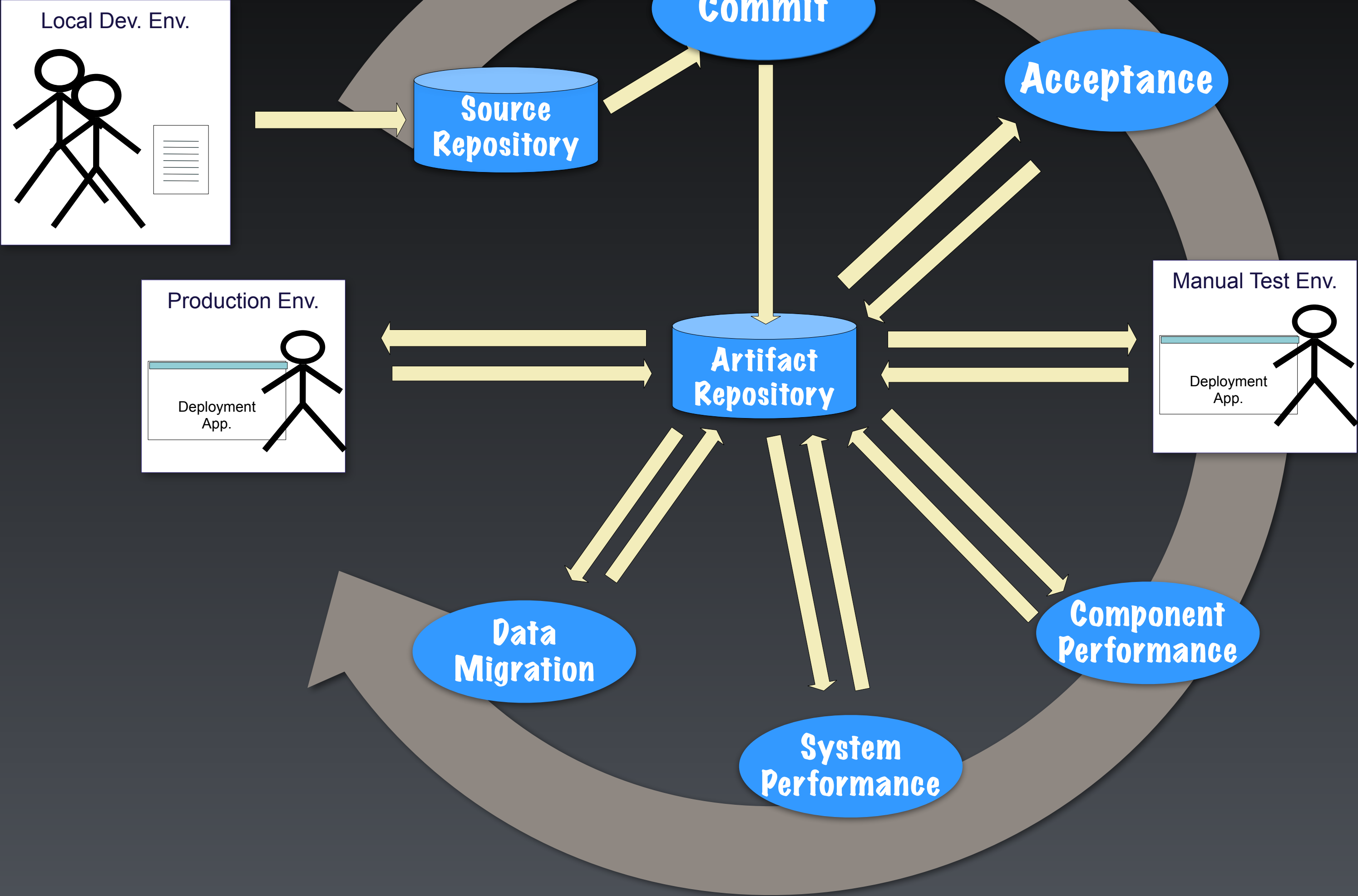
Deployment Pipeline



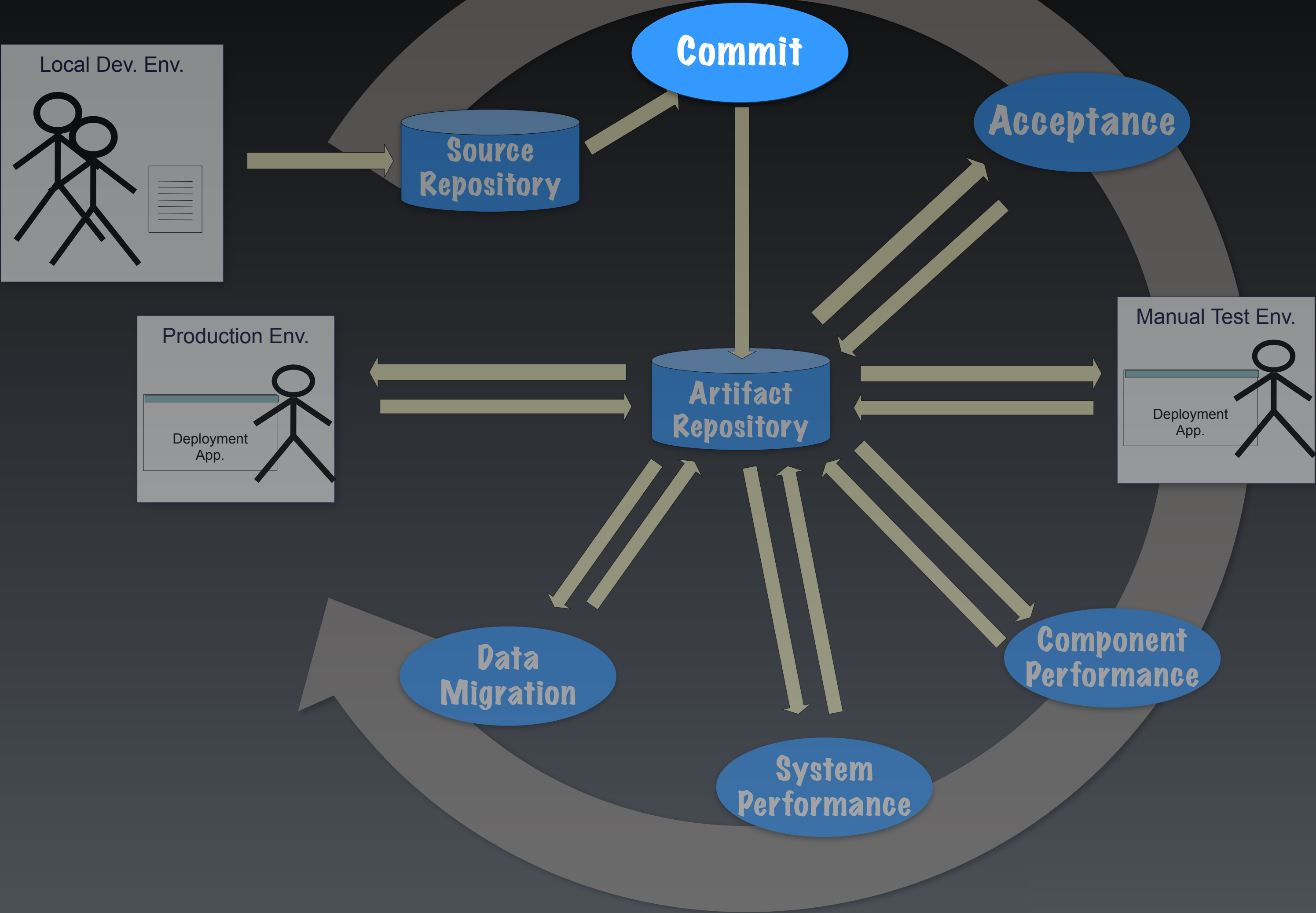
Deployment Pipeline



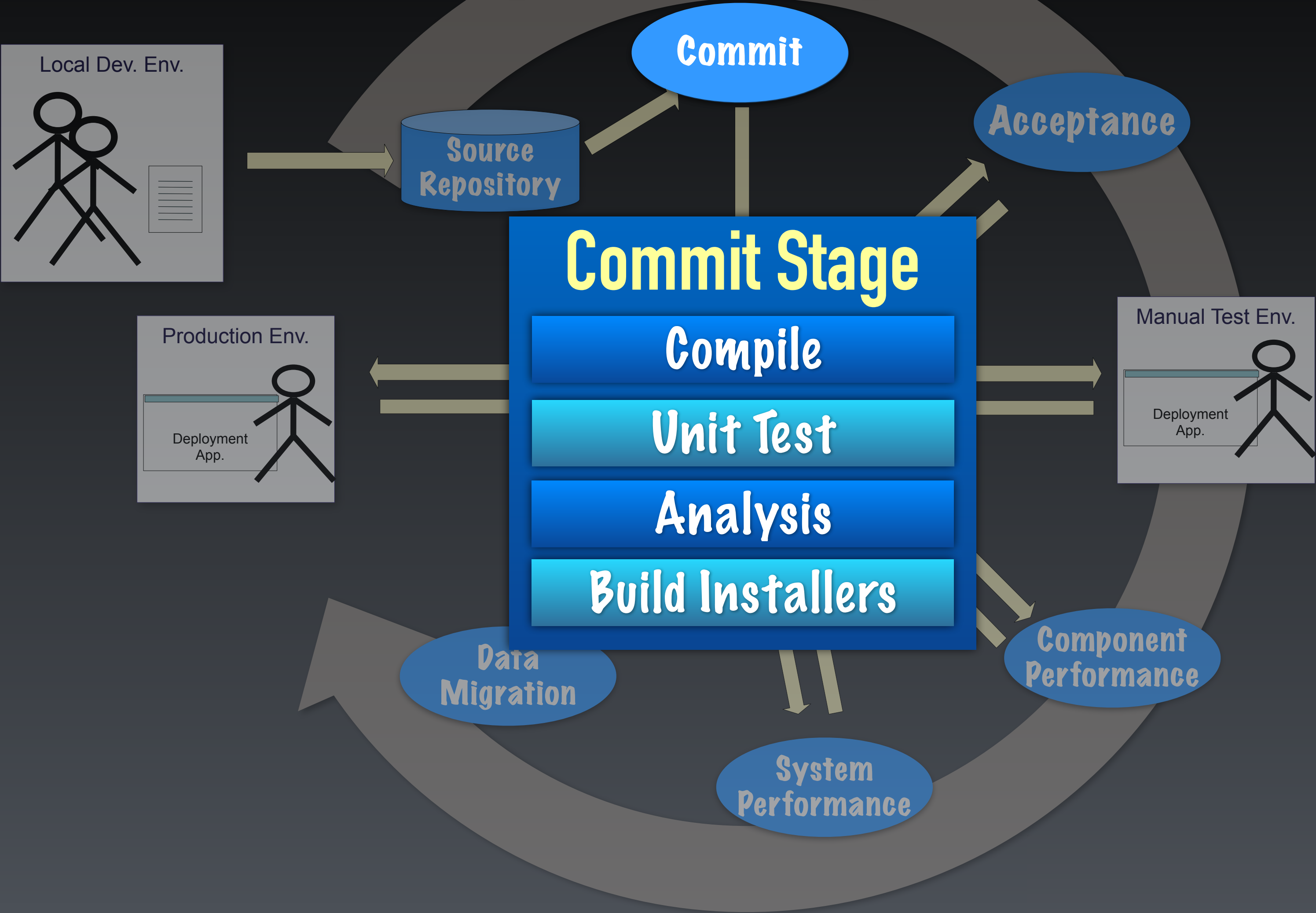
Deployment Pipeline



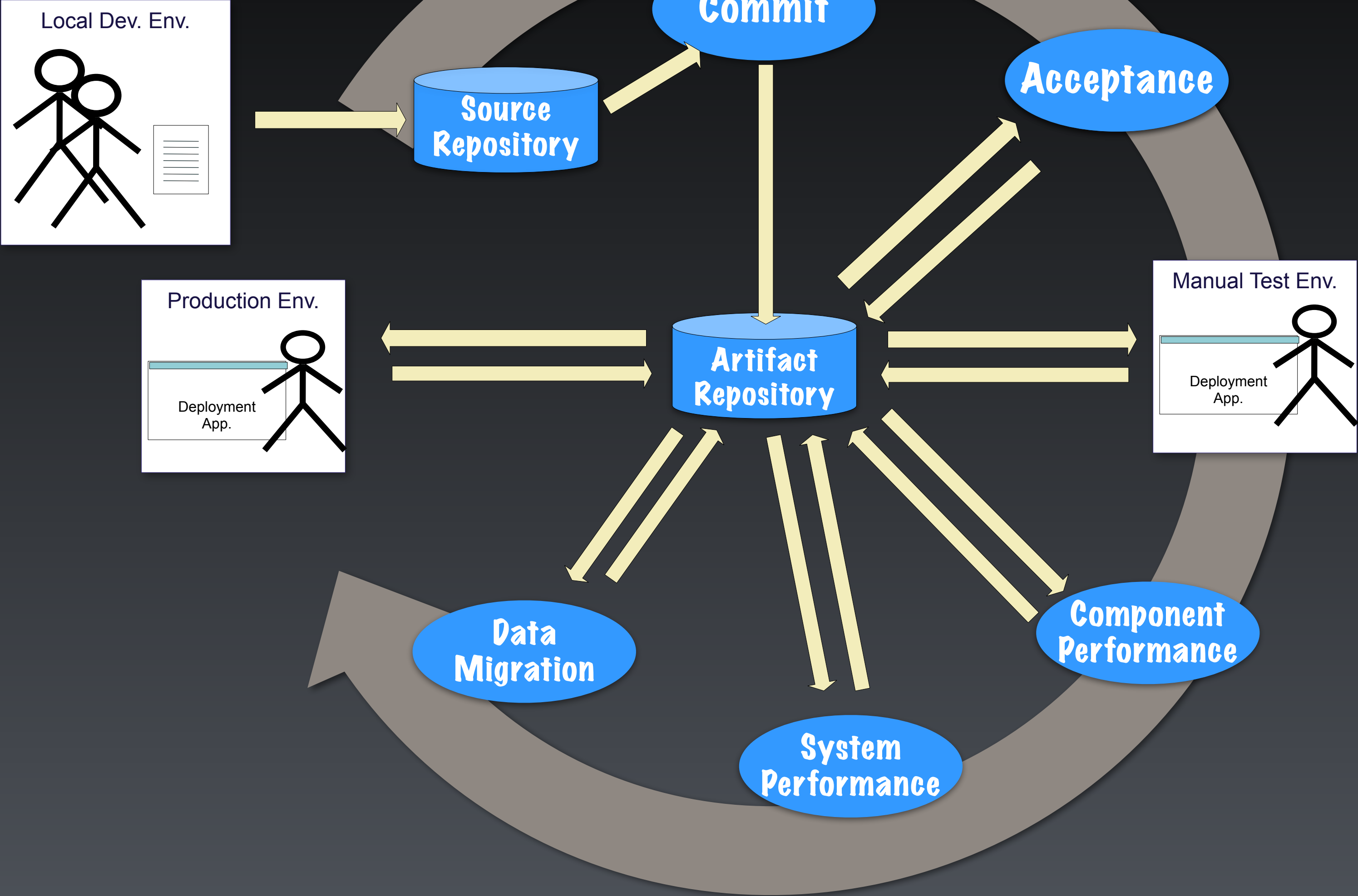
Deployment Pipeline



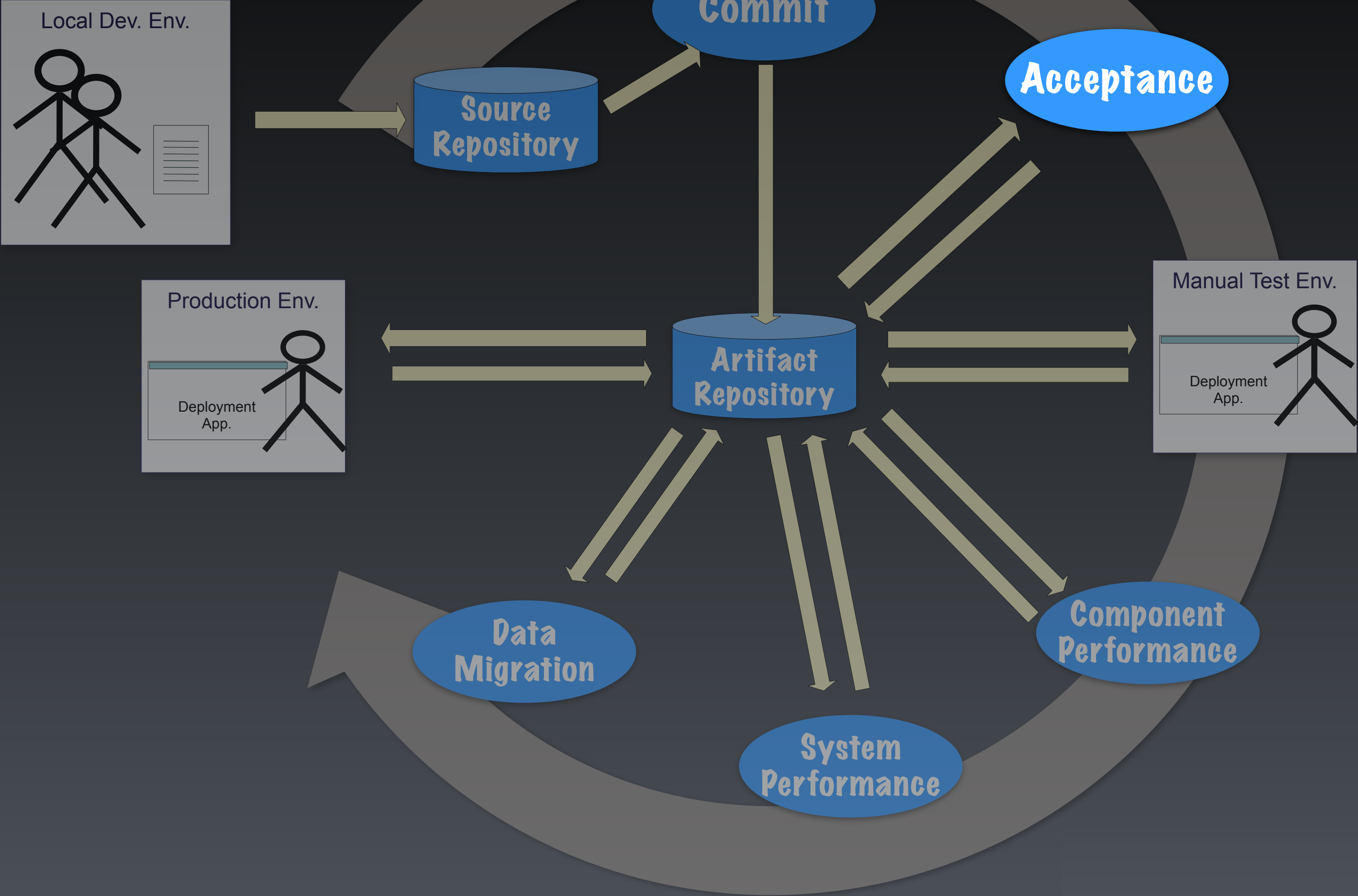
Deployment Pipeline



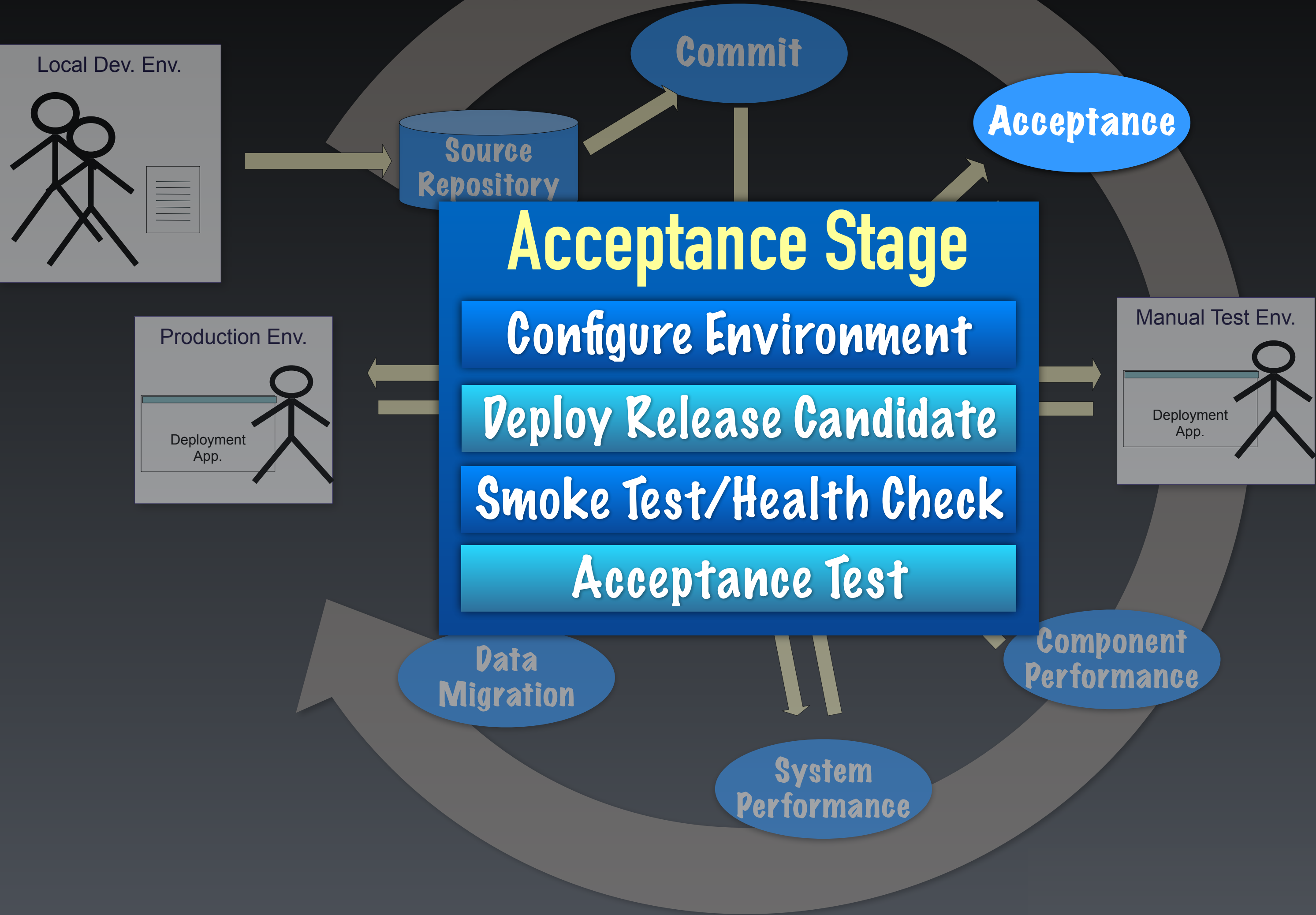
Deployment Pipeline

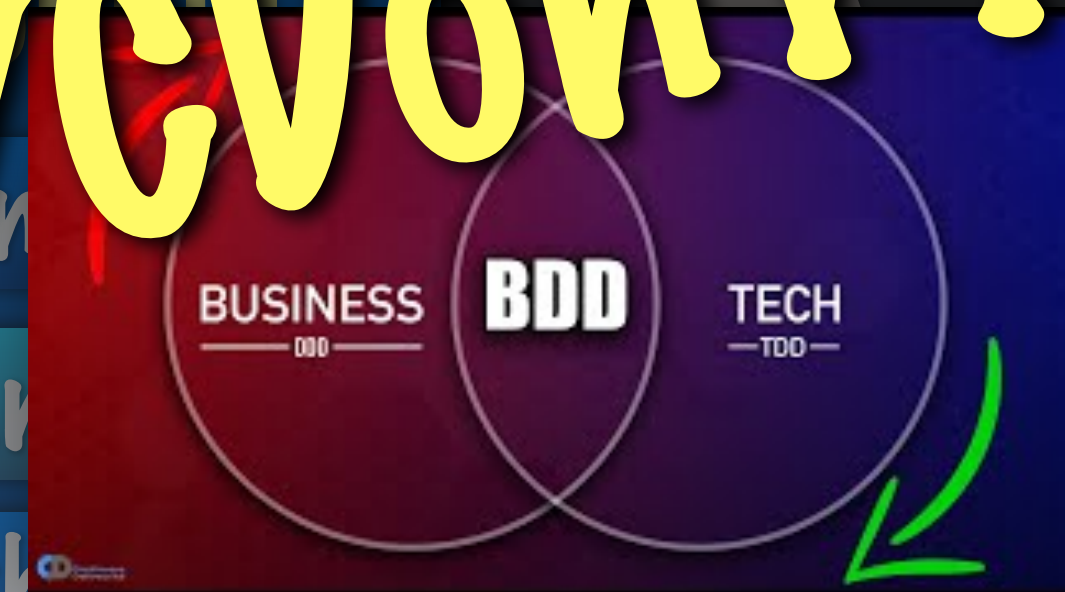
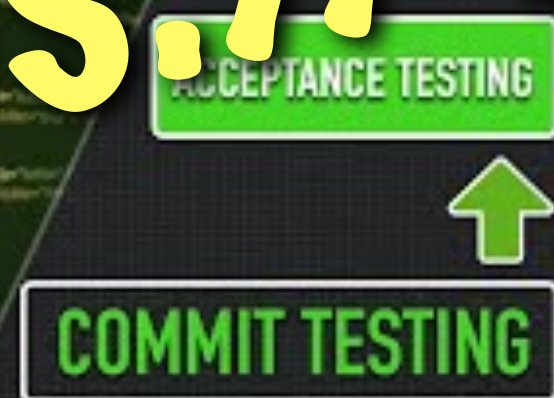
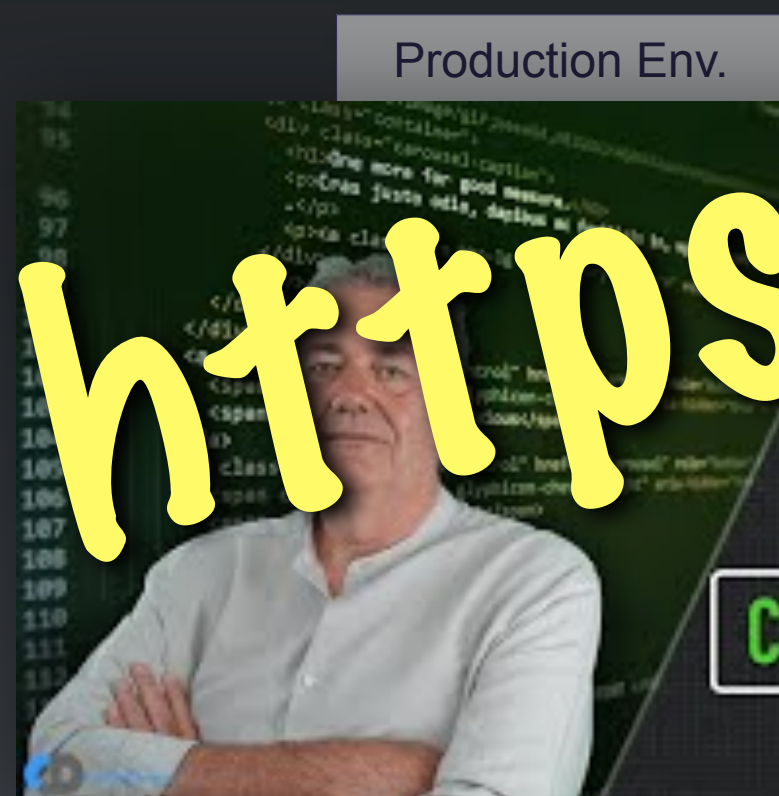


Deployment Pipeline



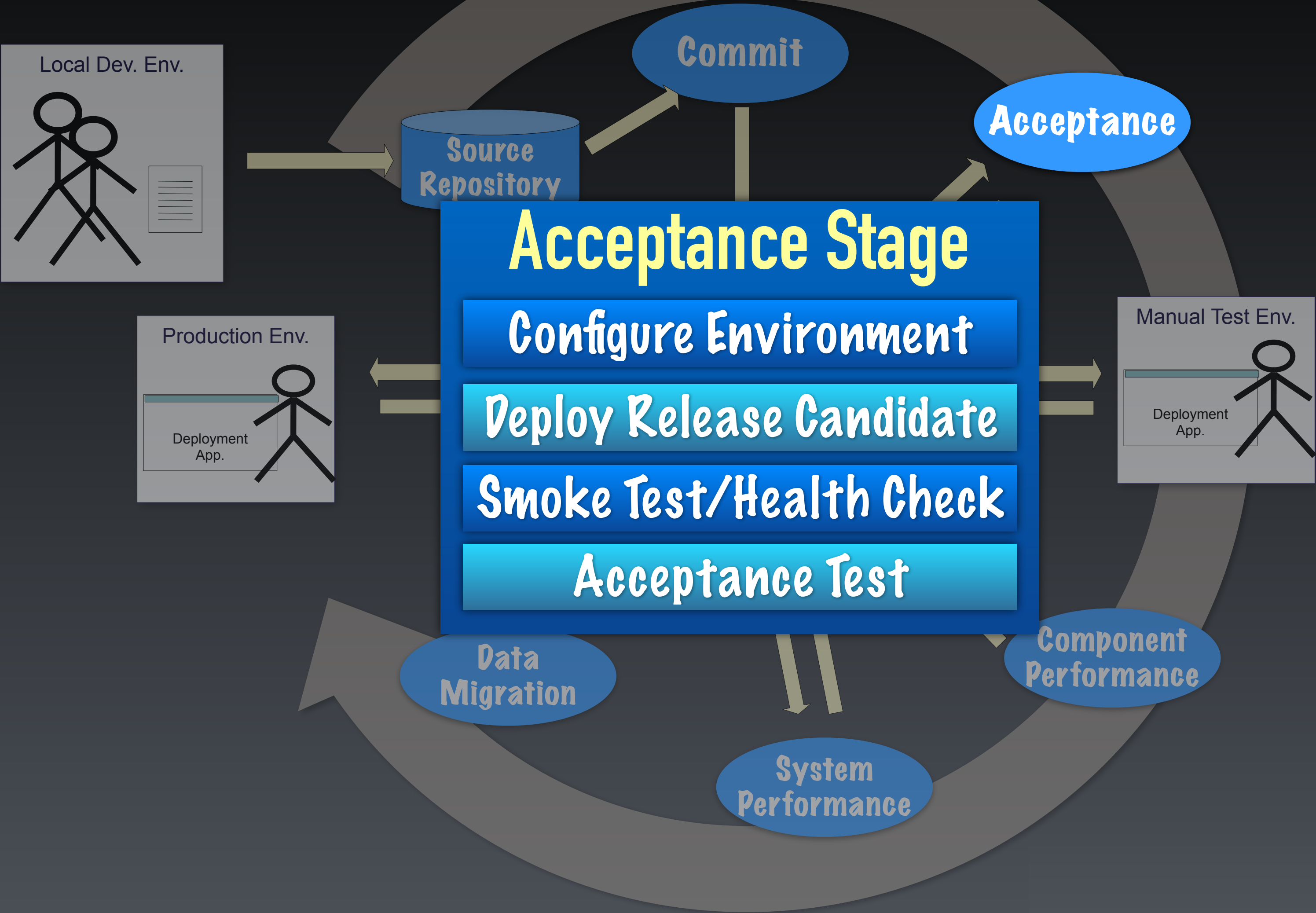
Deployment Pipeline



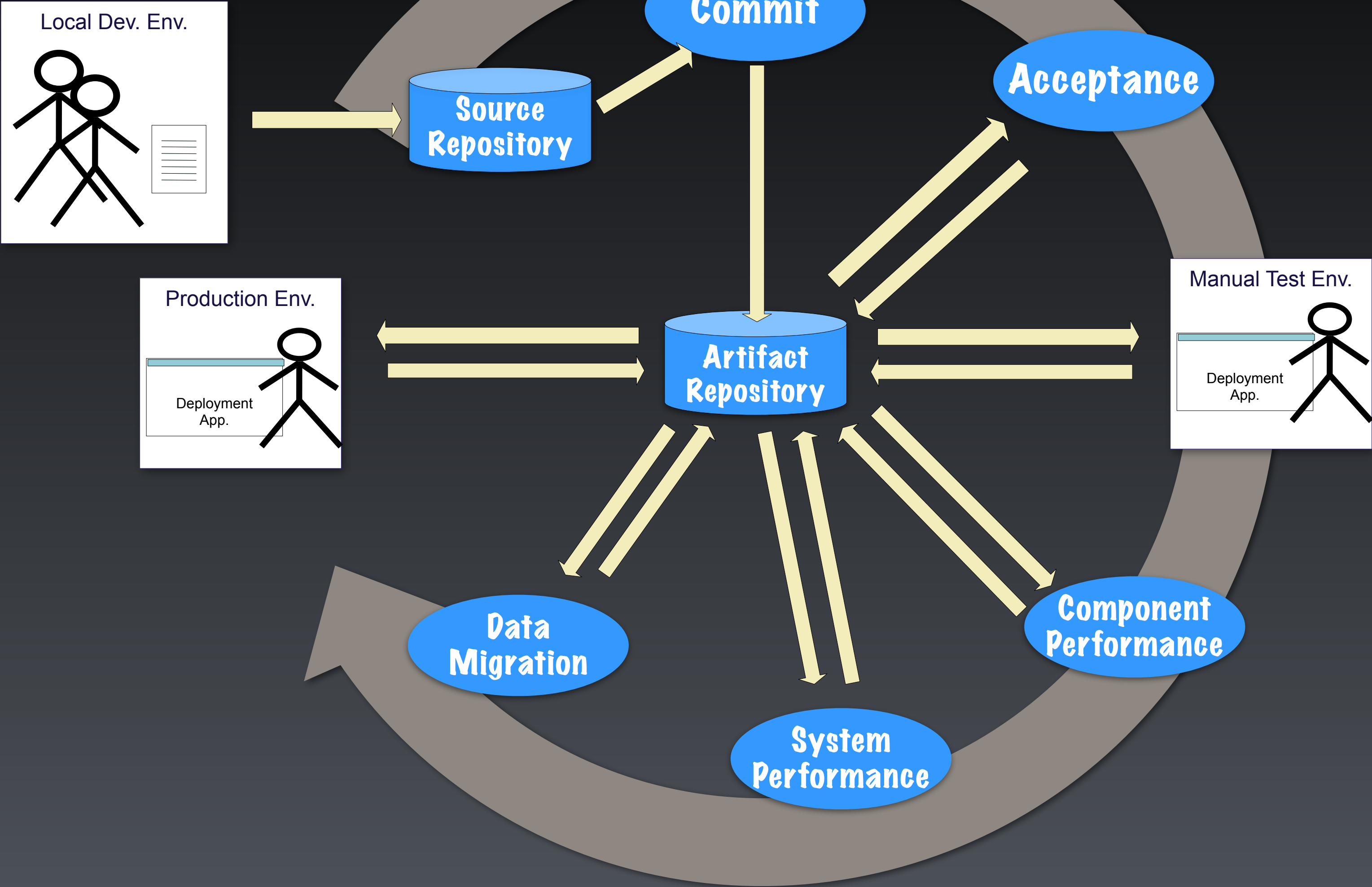


<https://bit.ly/cdonYT>

Deployment Pipeline

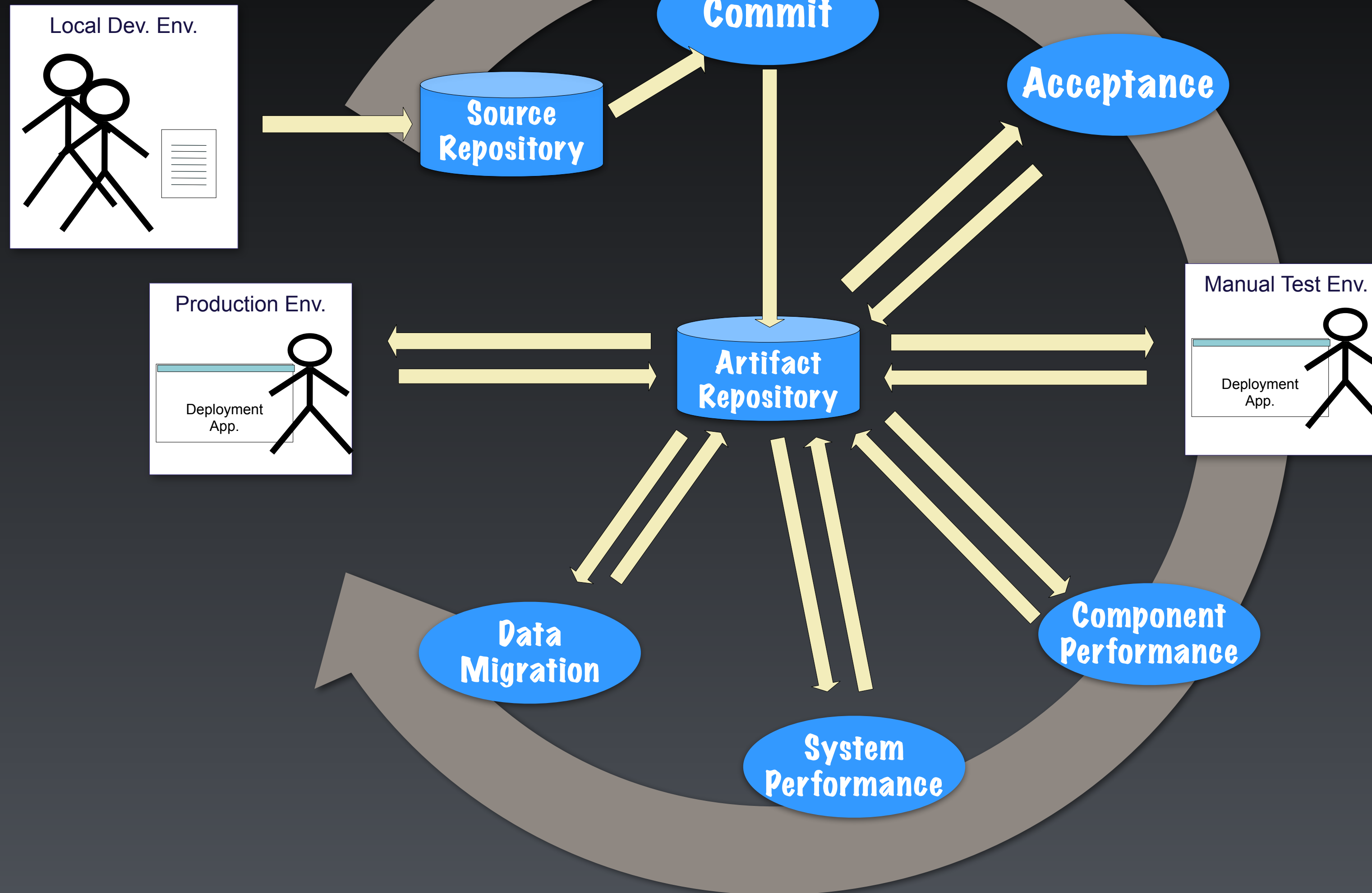


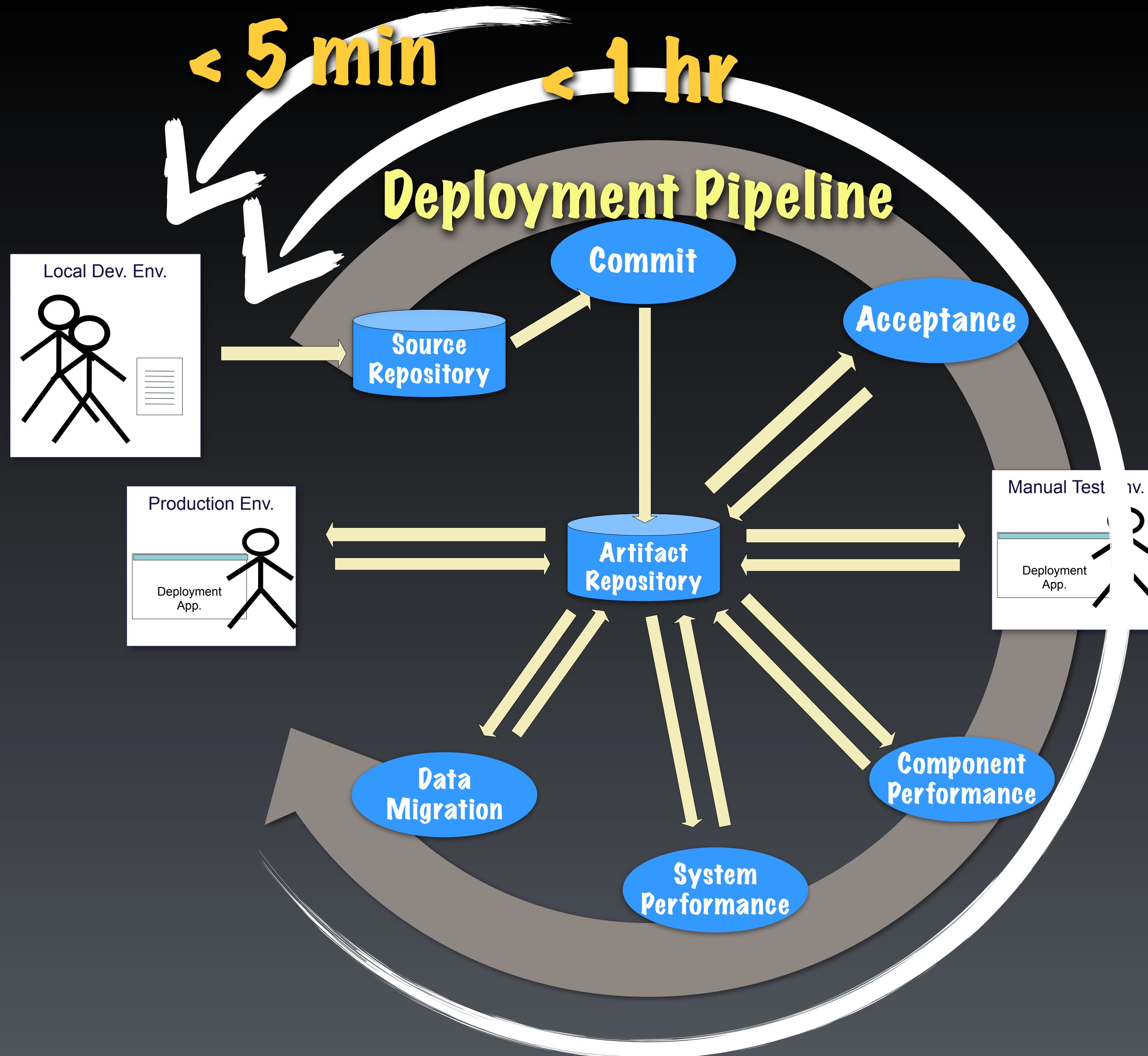
Deployment Pipeline



< 5 min

Deployment Pipeline





If This Were Real Engineering...
**It Would Improve the Quality &
Efficiency of Our Work!**

ELITE PERFORMERS

Comparing the elite group against the low performers, we find that elite performers have...



208
TIMES MORE
frequent code deployments

106
TIMES FASTER
lead time from
commit to deploy



2,604
TIMES FASTER
time to recover from incidents

7
TIMES LOWER
change failure rate
(changes are $\frac{1}{7}$ as likely to fail)



Throughput Stability

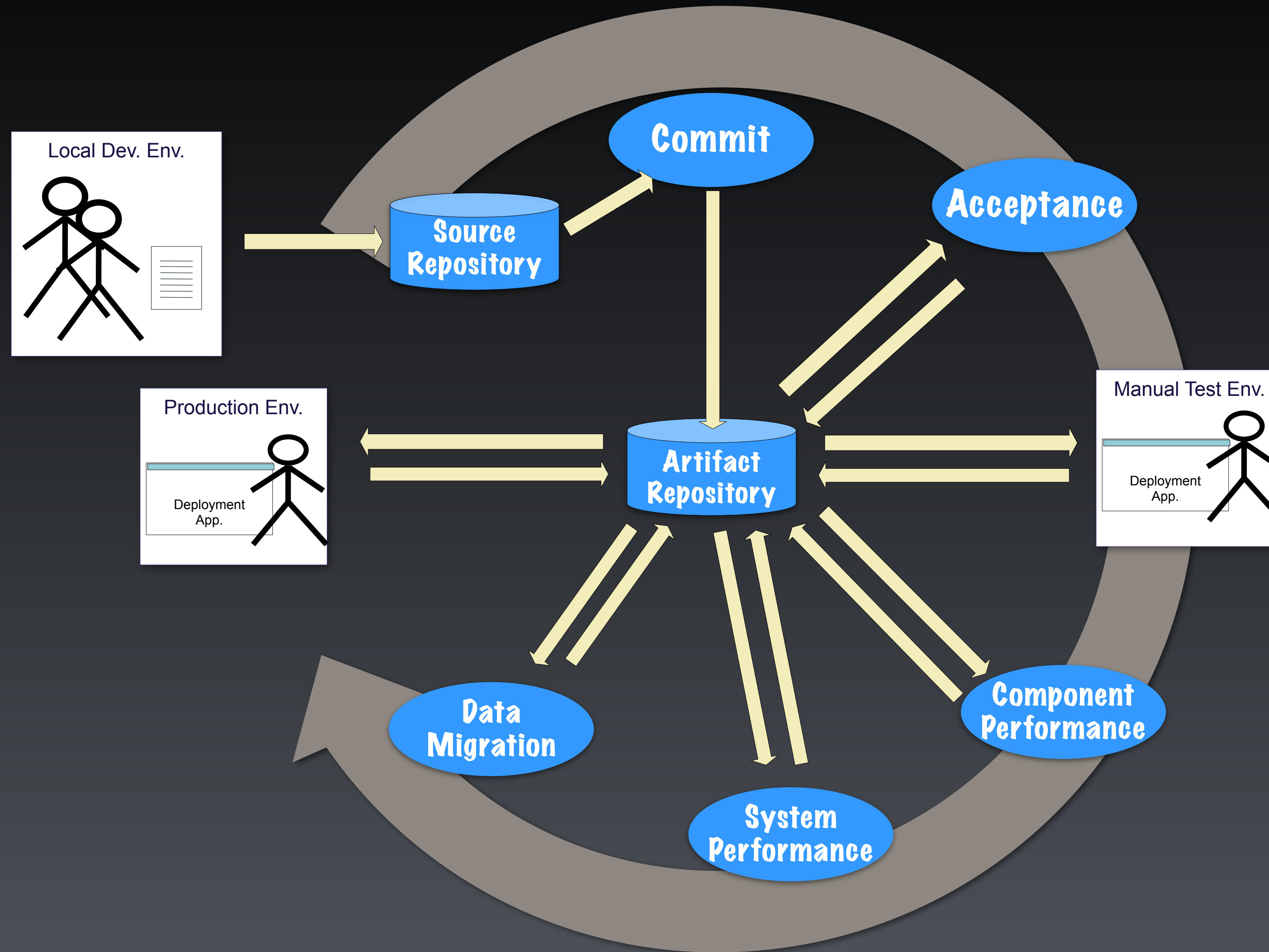
Better Software Faster!

ELITE PERFORMERS

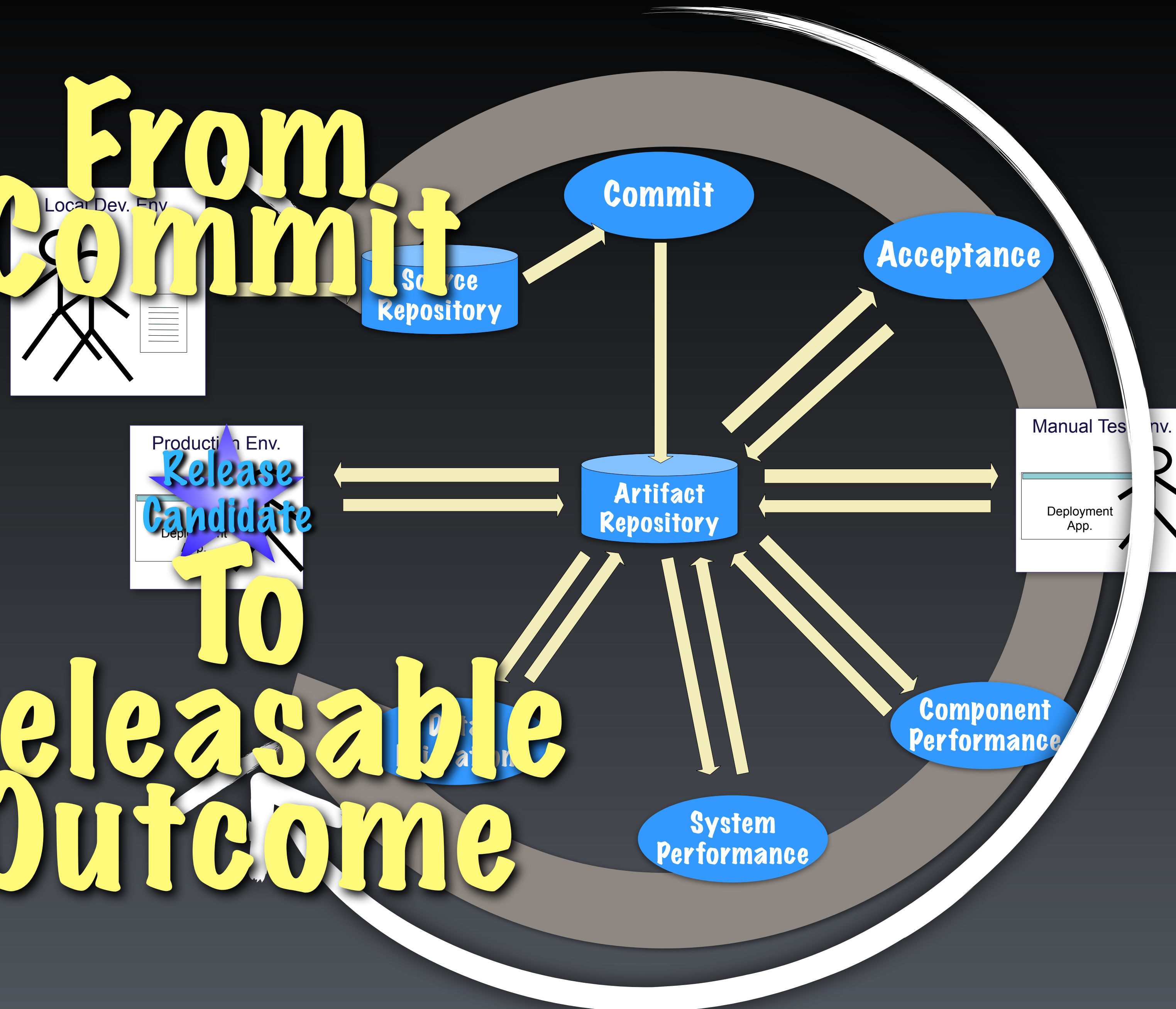
Comparing the elite group against the low performers, we find that elite performers have



Throughput Stability



From Commit To Releasable Outcome



Q&A



<http://www.continuous-delivery.co.uk>

Dave Farley

<http://www.davefarley.net>

@davefarley77

YouTube: <https://bit.ly/CDonYT>

