Engineering DevOps

to

Meet Business Goals

A White Paper
by
Trace3

January 2017

A growing number of enterprises, across many industries, are embracing DevOps as a central part of their digital transformation strategy. Given spectacular results reported by many businesses it is easy to understand why. The 2016 *State of DevOps Report*[1] indicates that the highest performing DevOps implementations are exceeding the performance of others across six enterprise business goal categories as indicated in the list below.

- **Agility**: 200X more frequent deployments, 2,555 times shorter lead times
- **Stability**: 24X faster recovery times for failures
- **Efficiency**: 22% less time spent on unplanned work and rework
- **Security**: 50% less time spent remediating security issues
- **Satisfaction**: 2.2X employees more likely to recommend their employer as a great place to work
- **Quality**: 3X lower change failure rate

Companies with high-performing IT organizations are twice as likely to exceed their profitability, market share and productivity goals.[8]

These are impressive results which every business would like to accomplish before competitors beat them to it. Yet only a minority of enterprises are achieving such high-performance DevOps implementations. Many are struggling to realize DevOps at all, at the level of business units and enterprise. This paper outlines an engineering approach for businesses and enterprises to implement DevOps, at the business or enterprise level, that meets specific business transformation goals in the fastest time with the least cost and without false starts. The approach described in this paper is like the concepts of value-stream focus described in the DevOps Handbook[7].

The first thing to understand is that DevOps is much more than a tool. You can't just buy a tool and have DevOps! There is a lack of a common industry definition of DevOps and a lack of a standard approach to DevOps. Despite the lack of agreement and standards, successful DevOps implementations share some common foundations. Per the "Seven Pillars of DevOps"[2] there are seven common practice categories which form the foundations supporting high-performance enterprise DevOps pipelines, as illustrated in Figure 1.

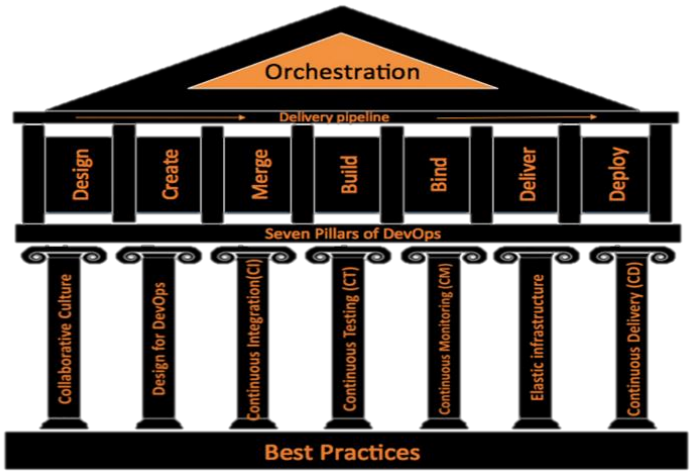

Figure 1: Seven Pillars of DevOps

The seven essential pillars for ensuring success with DevOps are listed below.

1. **Collaborative Culture:** Business goal alignment requires cooperative team culture. No amount of technology will provide effective end-to-end DevOps workflows if enterprise leaders, middle managers, Dev, QA, Infra and Ops teams do not cooperate with each other.
2. **Design for DevOps:** DevOps works for almost any product, but the best DevOps performance is achieved for products that are designed in modular fashion using service oriented architectures, microservices, 12-factor apps design methods and packaged as containers. That is not to say legacy products cannot benefit from DevOps, but modular, immutable architectures are most readily suited to take full advantage of DevOps.
3. **Continuous Integration (CI):** Software integration is engineered in accordance with best practices to merge changes quickly while minimizing rollbacks, interruptions and costly delays in the pipeline.
4. **Continuous Testing (CT):** Tests strategically conducted throughout the end-to-end pipeline provides relevant coverage to catch risky failures before production while completing tests quickly enough to avoid introducing bottlenecks.
5. **Continuous Monitoring (CM):** Real-time lifecycle intelligent active monitoring and analytics of tests processes and application performance measures are essential for real-time decision analytics at each stage of the pipeline to prevent analytics from causing bottlenecks.
6. **Elastic Infrastructure:** Infrastructures that are resilient and elastic to support on-demand vertical and horizontal auto-scaling perform much best for DevOps. DevOps applies to most types of infrastructures including private data centers, special purpose bare-metal systems, virtualized functions, containerized packaged applications, private and public clouds and hybrid cloud environments.
7. **Continuous Delivery and Deployment (CD):** Automated configuration management, application release automation, modular delivery packaging and deployment orchestration solutions with orchestration of virtualized and containerized applications are preferred over monolithic mutable applications.

As shown in Figure 2 the end-to-end DevOps pipeline is orchestrated and automated in stages from the initial work backlog through Design, Continuous Integration (CI), Continuous Delivery and Deployment (CD) and Live in-production stages. Collaborative culture, Continuous Testing (CT) and Continuous Monitoring (CM) and Elastic Infrastructures are active from end-to-end.

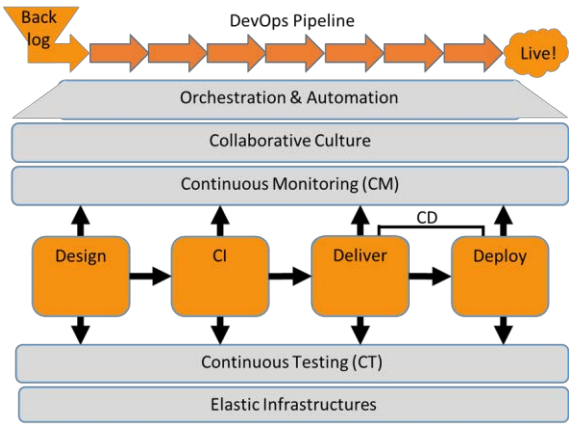

Figure 2: DevOps Pipeline

DevOps works best when all seven pillars are balanced to complement each other in a cooperative fashion. In other words, DevOps, does not work very well if some of the pillars are implemented well but others are not. For example, if CI and CD are implemented very well but the product is not designed for DevOps and the culture is not collaborative then DevOps may have some benefits but will not be accomplish the full potential and may not accomplish goals for the business.

An engineering approach is recommended for DevOps to accomplish specific business goals by being mindful of the need to keep all seven pillars in balance as DevOps matures along the journey towards higher performance. It is necessary to understand how DevOps works to understand how DevOps pipelines can be engineered to do this. The end-to-end behavior of a sequence of DevOps pipeline stages can be modelled as a multi-stage engineering process consisting of a sequence of processes, as shown in Figure 3. Each stage (Dev, CI, Delivery, Deploy) can be considered to have a change input rate (Xi/t), a processing time (Xt) and a change failure rate (Xf). Using this model the variables for each stage can be varied to determine the effects on then end-to-end time to process changes and the quality of deployments into production. Time and quality are the two dominant benefits that most businesses want to optimize first because the others (stability, security, efficiency and satisfaction) depend on the DevOps mechanisms which accelerate throughput while maintaining quality. An optimum ratio (OR) can be computed as the delivery rate divided by the defect rate. OR is highest when the change delivery rate is highest for the level of quality where quality is correlated to the number of defects/per change resolved prior to deployment to production.

The charts shown in Figure 3 indicate results from running the model with simulation software and a variety of input parameter variations. The charts show that optimal engineering of DevOps requires a balance of the all the stage parameters. The optimal performance (Highest OR value) occurs when the backlog rate (input changes per unit time) is highest, the # defects found per stage is balanced across the stages and the stage sizes are also balanced. Note: the data in the chart %Defects Found Per Stage Profile" shows % pass rate per stage, not # failed changes per stage. The "Stage Size Profile" data shows highest OR when variance of changes volumes is lowest. This balance effect indicates that continuous flow across the pipeline is optimal compared to alternatives in which each stage has varying processing rates. Using these principles, this model pipeline provides guidance for engineering real DevOps pipelines. Most importantly the model confirms that a balance of the pillars is critical to DevOps engineering for optimum results.

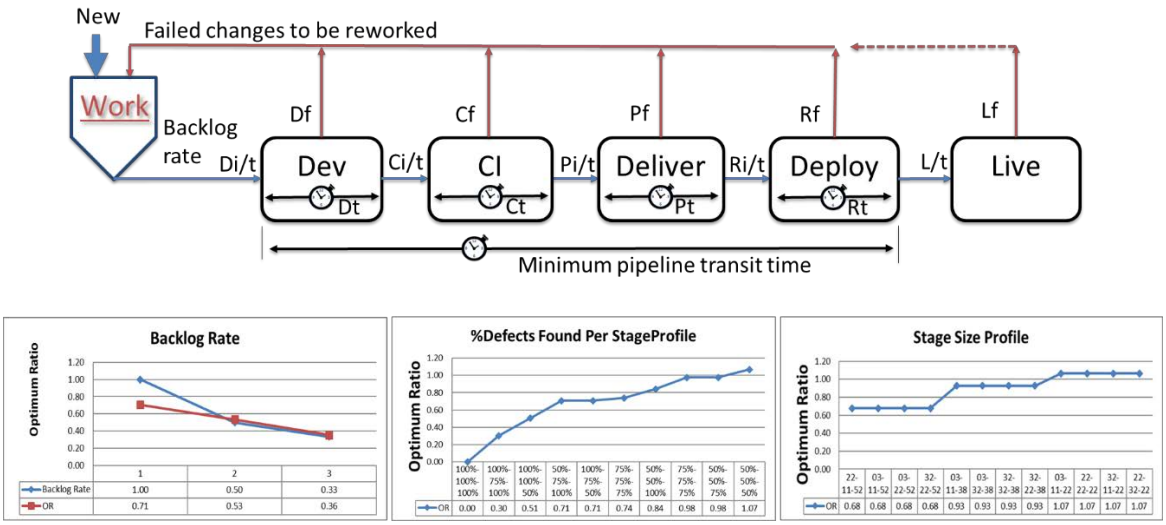

Figure 3: DevOps Pipeline Model

Therefore, to accomplish a balanced pipeline and the full potential of DevOps, a symphony of transformations across all seven pillars, illustrated in Figure 4, is the best approach.

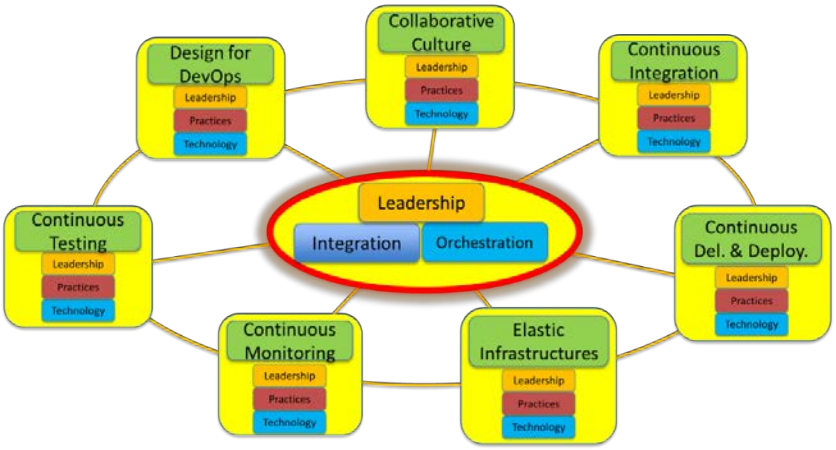

Figure 4: Symphony of DevOps Transformations

A symphony of seven parallel transformations is challenging to implement. What is the best way to start and co-ordinate these transformations? Strategic co-ordination is required! As indicated by the bubble in the center of Figure 4 strategic co-ordination requires leadership, integration and orchestration of systems. Leadership is the highest priority because leaders decide what are the systems that need to be integrated. After the systems are decided, then they can be orchestrated into an operational pipeline. Therefore, DevOps experts insist that for DevOps, culture and workflows are most critical.

A “Top-down/middle-out” approach to leadership is preferred over a “silo’d/organic” approach to leadership for DevOps transformations. Unfortunately, most organizations, in absence of enterprise level leadership, naturally tend to follow a “silo’d/organic” approach to DevOps. It is often easier to start initiatives and get agreement within a local business function or at the middle management level than to obtain enterprise-wide agreement regarding the goals for DevOps. However, this may be counter-productive without top-down strategic guidance.

With a “silo’d/organic” approach, local DevOps initiatives are undertaken in various parts of an enterprise. The efforts typically do demonstrate positive benefits for the local organizations. However, without an overall enterprise context these separate DevOps successes may create competing DevOps solutions. The very success of separate DevOps systems may create more silos and enterprise level friction and slows the realization of DevOps at the enterprise. The resulting internal competition can cause costly false-starts, a high % of failure at the enterprise level, and dissatisfied staff. A silo’d/organic approach typically will show positive results every few months, but some of the results will be counter-productive from the enterprise level. It is the experience of the author that it may take multiple years to accomplish enterprise-wide DevOps success with this approach.

A “top-down/middle-out” approach starts with the most senior leaders agreeing on enterprise level business goals and strategy for DevOps. This approach provides leadership alignment which can be translated into middle management goals. Once the leadership team including middle-management is in alignment it is much easier to establish collaborative teams. Local initiatives launched in the context of an enterprise strategy contribute directly to an enterprise-wide DevOps solution. Less than two years to enterprise-wide success has been reported by high performance companies that have undertaken a top-down/middle-out approach such as Capital One[6].

Enterprise level transformation, of any kind, requires active participation of business leaders. Successful leader participation requires courage, passion, investment, determination and measurement of progress. There are multiple dimensions to the transformation of leadership for DevOps. Most of them require changing culture, attitudes, beliefs and re-orientation of local systems into business level services. Successful transformations require mid-level managers to become collaborative facilitators. Organization silos must be re-organized into collaborative organization structures. Goals and reward systems need to change from functional to business level perspectives. Local processes, infrastructures and tools need to transform into business-wide services view instead. Training programs must change to emphasize cross-functional organization training. Local Key Performance Indicators (KPI) are changed into business level service level agreements (SLAs).

A comprehensive approach to DevOps transformation starts with a systematic envisioning process as shown in Figure 5.

Figure 5: DevOps Envisioning Approach

Leaders and key functional staff representing the breadth of business, Dev, QA, infrastructure, and Ops teams meet, in a workshop setting, to create and agree on the scope, business level goals and goal priorities. At the enterprise level, it may be decided at this point to focus on a specific business unit or product initially and then expand the solution to other business units as the solution is proven.

DevOps transformations and journeys at the enterprise level, typically span multiple years. It is critical for business level and mid-level leaders to determine a few near-term goals for the DevOps transformation to focus the work initially. The DevOps goals will clarify deliverables that will demonstrate and measure progress and justify continuing the transformation after the goals are met. The goals need to be a specific as possible. For example, an enterprise may decide that the highest priority goals for the first four-month leg in the DevOps journey is to reduce time-to-market by 50%, make workflows visible to all stakeholders, change the culture to one of collaboration and trust and reduce frequency of customer reported failures by 25%. The actual goals should emphasize current business priorities.

Note: A series of small accomplishments in a short time helps show progress and build confidence and management visibility to success. Each enterprise DevOps journey is different. Some organizations may start from culture changes, some may start at the Dev end, some may start in the middle of the pipeline, or Delivery end or infrastructure. The optimum starting point is determined based on the envisioning process which considers business priorities together with gap analysis and workflow analysis. Pick a first milestone which closes the biggest bottleneck first, then the next and then the next. For example, if an organization decides to start at the Dev end of the pipeline then the first milestone will likely include changes to the Dev process and product architecture. The next milestone could be to accelerate continuous integration to take advantage of the improved design.

Once the DevOps goals are clarified and aligned the next step is to assess DevOps capabilities and gaps relative to the goals and industry best practices. A broad DevOps assessment determines the Gap between the enterprise's current practices and industry best practices using a DevOps Gap assessment tool. Figure 6 illustrates some example best practices, formulated as questions, that are used to assess

gaps. The DevOps assessment tool covers best practices for all seven of the DevOps pillars to ensure that a comprehensive and balanced assessment is performed. For each practice a score is assigned that reflects how well the practices is being followed. The score needs to be reviewed and agreed to by the cross-functional team. The process of reviewing and agreeing on the score is a valuable way to obtain alignment between team members before proceeding to the next step.

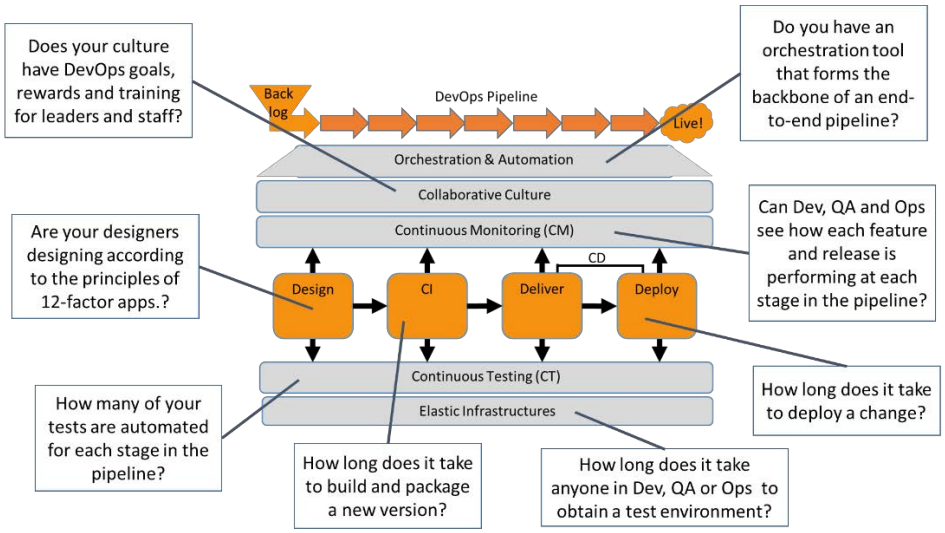

Figure 6: DevOps Gap Assessment

Once completed, the results of the DevOps Gap assessment are used to guide deep assessments into the areas that are have the most significant gaps relative to the goals. During the deep DevOps assessment, detailed end-to-end workflows diagrams are documented for all stages of the current pipeline, from concept creation through to live production. Figure 7 shows an example segment of a DevOps workflow assessment. Tools, infrastructure elements, deliverables and promotion metrics are documented for each workflow and stage in the pipeline. Attention is paid to areas that are bottlenecks or trouble-spots in the pipeline, especially areas highlighted by the Gap assessment.

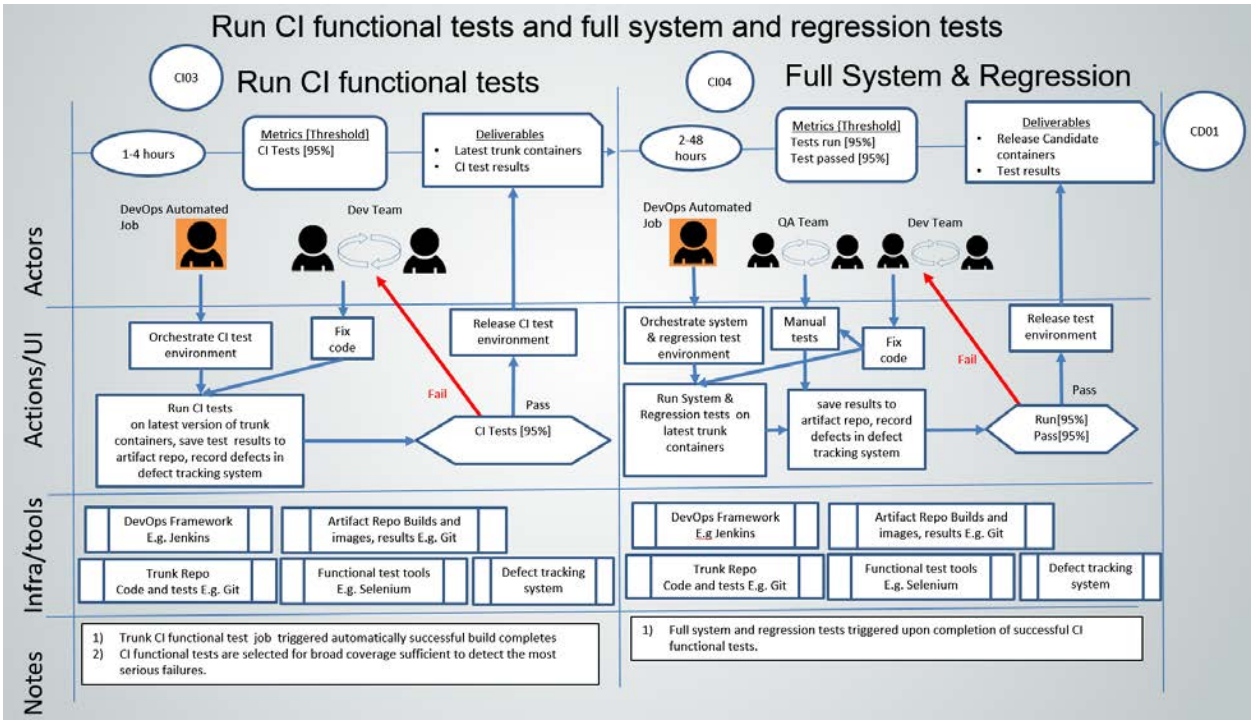

Figure 7: Example DevOps Pipeline Workflow Segment Analysis

With the goals, Gaps and workflow assessment data in hand, analysis can begin to formulate a DevOps solution and roadmap. The analysis process requires people with deep DevOps experience and expertise. The analysis process involves engineering a solution that satisfies the business goals by closing gaps and enhancing tools, infrastructures and workflows. This requires system level engineering knowledge. A strategic balance of tools, infrastructure, workflows, stage deliverables and metrics must be carefully engineered to ensure the resultant DevOps implementation will meet the business goals. By working backwards, from the end-to-end time and quality budgets, a budget for each stage within the pipeline can be set that will result in accomplishing the end goals. For example, if there is a goal to accomplish three delivery candidates per day, then each DevOps pipeline including stage development, CI, packaging and delivery need to operate at a throughput rate that will deliver 3 deliveries per day. To do that may require adding infrastructure and automation to support more parallel processing to speed up areas of the process that currently take too long. Similarly, to satisfy a goal to reduce customer reported defects may require introducing new test methodologies and test resources to relieve bottleneck points in the current workflow to improve test coverage. The additional testing will need to be fast enough to meet the new time budgets for the end-to-end pipeline.

Below is a step-by-step approach for re-engineering a DevOps pipeline to align with business goals.

1) Using the timing goals, determine a time budget for each stage that will equalize the stage timings and meet the end-to-end timing goal. (Total time/# stages)
2) Set a failure budget for each stage such that each stage fails at least 20% more than the subsequent one. ($Failure\_Rate^{Stage(X)} = 1.2 \times Failure\_Rate^{Stage(X+1)}$)

3) Set the input rate equal to the goal release rate increased by the sum of the expected cumulative failures through the pipeline.
4) Identify strategies to shorten the stage times that will meet the end-to-end time.
5) Identify strategies to increase defect find rates higher in earlier stages to match the budgets.
6) Rework the work-flows to match the new time and defect detection budgets.
7) Identify team, process, tools and infrastructure changes.
8) Set a milestone and measurements to demonstrate success for the milestone.

Table 1 is a summary of tactics that can be used to engineer each DevOps stage by adjusting elements of the seven DevOps pillars. The entries in the table are samples and not a complete list. It is best to engage experts that have a complete understanding of all options and interactions relevant to meeting specific DevOps goals within a balanced budget.

DevOps Business Goal	Tactics for DevOps Pillars (Examples)
Agility: fast delivery rate	• Orchestration and automation • Source control • Automate stage promotion decisions • Stop upon verdict failures (i.e. Virtual Andon cord) • Horizontal scaling - process modules in parallel • Vertical scaling - process each module faster • Dynamically process per relevancy of changes
Stability: fast recovery from failures	• Short circuit failures • Priority for stability improvements • Blue-Green deployments
Efficiency: save labor and capital costs	• Reduced corrective work saves non-value add labor • Elastic infrastructures enable sharing of resources
Security: reduce security failures	• Static analysis security checkers • Do not put credentials in automation scripts • Pen and DDS security test cases in functional & regression tests • Security test all production variations of OSs and browsers • Auto-audit production node for latest security patches • ACLs for pipeline stage artifacts & multi-tenancy labs
Satisfaction: improve collaboration and employee loyalty	• Leaders clarify business goals and results • Redefine or create new roles as needed to fill gaps • Empower staff • Training for DevOps practices and skills • Awards and incentives for collaborative successes
Quality: reduce failure rate in production	• Increase test coverage • A/B testing, Canary testing • Auto-scale test environment • Improved monitoring tools

Table 1: DevOps Tactics

After engineering the pipeline stages to satisfy business goals the resulting architecture can be used guide selection of infrastructure and tools components. Figure 8 is an example architecture and shows some tool choices. The onboarding of new tools and infrastructures needs to be controlled. Ticketing and workflow systems such as ServiceNow™ are very useful for this. Once the tools and infrastructure components are on-boarded, the software changes that are processed through the DevOps pipeline can be controlled using workflow management tools such as Jira™.

Once in place, DevOps evolution becomes a process of continuous improvement. Continuous improvement feedback, when used properly, will provide data to continue to accelerate and balance pipeline parameters. As change rates increase (faster is better per the model) the parameters across the model need to be adjusted to keep the entire pipeline in balance.

In cases where there is a large pile of technical debt, an assessment and strategy to pay down technical debt must be included in the DevOps implementation plan before forward progress can be made or else there is a risk that implementation plans will be interrupted for higher priority work on specific technical debt items.

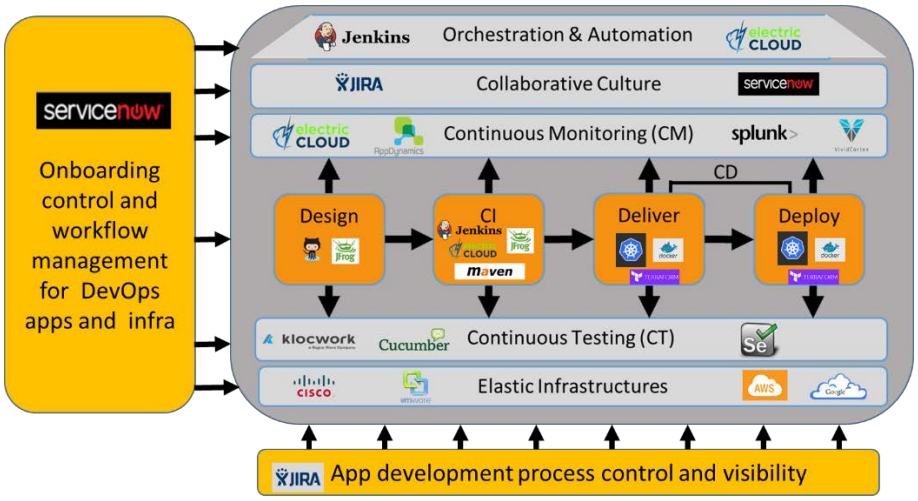

Figure 8: DevOps Example Architecture

Summary:

In summary, this paper described an engineering approach for DevOps to satisfy specific business goals. A balanced approach that addresses all seven DevOps pillars is recommended. A "Top-down/middle out" approach to leadership ensures goal alignment and tends to accomplish enterprise DevOps faster than Silo'd/organic approaches. An envisioning process identifies gaps and workflow deficiencies. Once the goals and current-stage data is documented, then a solution can be engineered to meet specific business goals. Once in place DevOps evolution becomes a process of continuous improvement. Continuous feedback provides data to continue to accelerate and balance DevOps pipeline parameters.

References:

1. State of DevOps Report, June 2016, published by Dora and Puppet Labs
2. https://devops.com/7-pillars-of-devops-essential-foundations-for-enterprise-success/
3. https://devops.com/can-culture-detract-success-devops/
4. https://devops.com/design-devops-best-practices/
5. https://devops.com/continuous-can-integration/
6. DevOps at Capital One, http://www.slideshare.net/ITRevolution/does-sfo-2016-topo-pal-devops-at-capital-one
7. The DevOps Handbook, October 2016, by Gene Kim, Patrick Debois, Jez Humble, John Willis
8. State of DevOps Report, June 2014, published by Puppet Labs

About Trace3

As a Transformative IT Authority, Trace3 is the premier provider of IT solutions. We integrate IT products and services with insightful consultation to provide total transformation for both executives and organizations. Our elite engineers implement tomorrow's systems and hardware to solve today's most pressing IT problems, standing shoulder-to-shoulder with our clients to protect and serve their interests.

About the author

Marc Hornbeek is Principal Consultant - DevOps at Trace3. Marc has 38 years' experience architecting, designing, developing and managing high-performance solutions for IT and engineering infrastructures deployed in commercial and government applications globally. Marc has served as CEO, Board Member, founder, corporate executive, CTO, VP, General Manager, Principal Consultant, Senior Solutions Architect and Professional Engineer. He has held key roles at Bell-Northern Research, Tekelec, ECI Telecom, GSI Lumonics, Vpacket, EdenTree Technologies, Spirent Communications and Trace3. Marc has been the innovation lead over many successful automation, Lab-as-a-Service and DevOps projects for systems manufacturers and operators. Marc is a regular speaker, blogger, author and educator on topics including DevOps, Lab-as-a-Service and continuous test automation. Marc is an author for the DevOps Institute, DevOps Continuous Delivery Architect course and the DevOps Test Engineer course. He is a 41-year life member of the IEEE and has been awarded the "2016 Outstanding Engineer" by the IEEE Region 6 - Western United States.