Digital Transformation with the Node.js DevOps Stack

PayPal, Netflix, and Walmart show the way to do rapid digital transformation of legacy systems

Why Digital Transformation

Modernizing systems and processes has become a top priority for businesses across all verticals. In simplest terms, digital transformation is "the use of technology to radically improve performance or (business) reach."[1] For businesses from established, traditional industries such as financial services, insurance, retail, and healthcare, digital transformation requires adopting a whole new focus, learning to "win on the basis of superior digital capabilities and to harmonize those capabilities with offline operations."[2] But for digital-native businesses that have never needed to make that shift — e-commerce, social media, web-based services — it involves a sharpening of that focus combined with an ongoing evolution of process and infrastructure.

> *Node.js is emerging as the de facto choice for companies looking to build the apps to achieve greater agility and drive Digital Transformation.*

In either case, organizations are looking for the fastest and most effective route to modernization; Node.js is emerging as the de facto choice for companies looking to build the apps and other infrastructure needed to make such a move. While building out services with Java will generally take anywhere from 8-24 months, Node.js teams typically get the development work done in 2-8 months. That's a massive cost and time savings. Node.js enables organizations to leverage resources they already have to replace monolithic solutions with more flexible, modular solutions.

As outlined below, Node.js facilitates a smooth and seamless implementation that is transparent to the user. Using Node.js, your business can incrementally deploy applications to replace the functions that make up the existing system—while introducing new functionality—until a new system completely replaces the old. Node.js also supports modern software development and delivery implementation methods that can transform how a business operates.

Before You Start: Build a Core Team (Start Small)

Getting started with Node.js requires assembling a core team and choosing an initial project scope. The core team will bring together frontend and backend talent, including members with JavaScript expertise and members with server-side experience—as well as legacy system subject matter experts (SMEs). It doesn't matter if your server-side experts

1 "Digital Transformation: A Road-Map for Billion-Dollar Organizations," Capgemini Consulting: http://nsrc.io/2brZ726

2 "Digital Business Strategy," Forrester: http://nsrc.io/2brZklY

are experienced Node.js users, or if they have never seen Node.js in action. It's the shared perspective that matters; bringing them together provides a complete picture. DevOps and Operations teams also need to be engaged on the core team from project inception.

When an Ops team member is not available to join the core team, a member of the core team should be designated as an Ops specialist and liaison. Likewise, when there are no legacy system SMEs available to serve on the core team, another member of the team will need to get up to speed with each legacy system involved and assume that role.

The process of digital transformation must be coupled with modern DevOps practices such as Continuous Integration and Continuous Deployment (CI/CD) to maximize agility. Delaying end-to-end integration introduces barriers to iteration and feedback. Implementing these processes after the project development lifecycle has begun creates an unnecessary "waterfall" phase; development teams have nothing to do while operations teams complete essential infrastructure tasks.

CPU BOUND TASKS	I/O BOUND TASKS
Time to complete a task determined by a CPU processing speed	Time to complete a task determined by wait time for inputs and outputs
EXAMPLES	**EXAMPLES**
• Rendering	• Get a user session
• Data Processing	• Query Database
• Natural Language Processing	• API Request

Like any tool, Node.js works better for solving some problems better than others. A good gauge for how well Node.js will apply to a particular task is to determine whether that task is CPU-bound or I/O-bound. As a rule, most service system tasks are I/O-bound or directly impacted by I/O-bound activities. CPU-bound tasks can be performed in Node.js, but the mixing of I/O-bound and CPU-bound tasks in the same process can lead to poor performance.

Node.js was developed for building performant network applications and excels at tasks tied to managing user interactions and other inputs/outputs. For example, Node.js is well-suited to rendering a dynamic web page that supports multi-query, asynchronous workflows. Node.js serves as the glue, making it easy to bring all the parts together — session data, data from database, user data — to enable seamless querying with a smooth and straightforward user experience.

Netflix is a great example of how effective Node.js can be for this kind of challenge. Traditionally an enterprise Java shop, Netflix transitioned all of its user interfaces to Node.js. While retaining the core of the Netflix engine in Java, Node.js has enabled Netflix to significantly cut build times and streamline the overall development process. Node.js also makes it easy for Netflix to tune its delivery of the user interfaces in ways that were just not possible with Java. In migrating from Java to Node.js, while simultaneously moving from the data center to the cloud, Node.js has enabled Netflix to adopt service-oriented architecture that matches their cloud-native deployment and enables teams to rapidly iterate.[3]

Start with Proxy-First Development

An important early consideration is the question of modularizing the system. Node.js is well-suited to modularization, which is often implemented through a microservices architecture. If you can structure the digital transformation process such that you will chip away piecemeal projects from another technology that is slow to change (Java, Perl, etc.), then your path for modularization is pretty clear. On the other hand, if the project calls for an initial proof of concept with an elastic scope, modularizing might not be the best approach. Successful digital transformation most often occurs when companies start with smaller slices of a legacy application, adopting new approaches quickly when there is a clear benefit.[4]

The key to incrementally replacing the existing system via modular buildout of a new system is proxy-first development. For each function that you want to replace, you build a simple Node.js app that proxies a request from the outside to the existing service that it will replace. Ryan Dahl, the creator of Node.js, has said that successful Node.js architectures are "proxies all the way down."[5]

If your project is a good fit for a modular approach, the best way to begin is by adapting a single discrete task, even if it is relatively small. Using Node.js, you develop a proxy for that task within the working solution, building a cross-functional experience with the Node.js platform.

To implement the proxy, you'll first need to establish an API gateway to serve as traffic cop. This is also a proxy, but it does not necessarily have to be written in Node.js. With the API gateway in place, your usual traffic management solution can govern the operation of the proxy within the larger system's framework. The initial versions of the proxy mirror the existing data structure passing through the system. As new services are deployed,

3 Kiran "CK" Oliver, "How Node.js Powers the Many User Interfaces of Netflix," TheNewStack: http://nsrc.io/2bs06zx (December 3, 2015).

4 "Digital Transformation Using Node.js," Forrester: http://nsrc.io/2i7zjah

5 Dan Shaw, "The Engineering Case for Node," Github: http://nsrc.io/2b6WcXK (November 16, 2014).

there is often an opportunity to optimize the structure of the data being sent compared to the legacy system.

Three Types of Proxy to Support Digital Transformation with Node.js

API GATEWAY PROXY	(STARTER) PROXY APPLICATION	DATA TIER PROXY
Determines what services respond to a network request	Technically a reverse proxy, which means it makes requests on behalf of another app.	The service that abstracts away what data at rest is, defines how you interact with data rather how store the data.

The basic approach is to start small with a proxy app and modify it as you expand its use. Build that proxy app using your Continuous Integration (CI) processes, and create a container that is then deployed using your Continuous Deployment (CD) pipeline. This replicates your production environment, enabling cross-functional engagement (and collaboration) and establishes the toolset you need for rapid iteration. You're ready to test the solution.

This approach provides the advantage of achieving a quick win by putting part of the solution directly into production, starting you on the path to digital transformation with Node.js. When you go live it isn't just with the developers; you go live with Ops.

PayPal is a great example of major transformation within an inherently digital business. PayPal started its Node.js implementation by focusing on a single task, carving off a single discrete piece of functionality from the existing monolith. In this instance, the task was the account overview page — one of the site's most important and heavily used assets. From the limited scope of that first project, the Node.js team within PayPal began to iterate.

> *By taking this rapid and incremental approach, PayPal replaced its massive legacy environment with a modern, fully-modularized service-oriented architecture.*

After implementing the accounts overview, they tackled the user login experience, and then the main PayPal.com landing page. In all, they repeated the process of replacing modular service components some 50 times, iterating and improving as the project team's Node.js expertise grew.

By taking this rapid and incremental approach, PayPal replaced its massive legacy environment with a modern, fully-modularized service-oriented architecture. Even digital-native organizations like PayPal need to accelerate the pace of innovation. Node.js has enabled the next generation of applications to evolve rapidly to respond to the needs of an ever changing market.

Transform the Software Development Process

Node.js supports modern software development and delivery methods that better address the challenges that businesses typically face today. Standard development practices of the late 1990's and early 2000's produced applications deployed as monolithic frameworks, most often built in Java and resting on a vast base of Java code. Such environments lend themselves to a fairly slow development and implementation cycle. In recent years, a revolution in software development has replaced that monolithic approach with unintrusive, iterative methodologies that encompass relentless incremental development via frequent releases. Designed for the modular approach, Node.js supports simplifying and streamlining development processes to align with these methodologies.

Walmart went through an interesting digital transformation journey while building out mobile services on its legacy Service Oriented Architecture (SOA) environment. Their existing, massive system was simply not architected for mobile. For example, making calls to certain data warehouse services could take seconds or even minutes. Additionally, many legacy SOA services returned XML, while modern mobile apps required JSON. Overall, the legacy SOA architecture was completely unacceptable for mobile applications. The constraints and requirements for these downstream data services need to be be understood, measured, and in many cases, transformed.

Walmart found the perfect solution to these challenges in Node.js, which has enabled support for "end-to-end Javascript."[6] With Node.js, Walmart can now deliver complex functionality in an efficient and lightweight development loop. Node.js has also enabled Walmart to implement a new, modern software delivery model to complement the new architecture and apps. The development team provided a vivid demonstration of this when they proceeded with a scheduled software release… on Black Friday. The biggest shopping day of the year (and the busiest day, by far, for a company like Walmart) was no hindrance. The team deployed the new release while 200,000 users were online.[7]

Accelerate the Transformation

After implementing your first task on Node.js, and kicking off the iterative process, there are other important steps to take toward digital transformation. For example, it

6 J. O'Dell, "Why Walmart is using Node.js," VentureBeat: http://nsrc.io/2boBenz (January 24, 2012).

7 Cian Ó Maidín, "Why Node.js Is Becoming the Go-To Technology in the Enterprise," nearForm: http://nsrc.io/2b71baN (March 10, 2014).

is important to establish a series of benchmarks between apps developed in Node.js and the corresponding pieces of the old solution. Common data points to examine are response times and cpu/memory load. Often teams discover the hardware requirements are lower when utilizing Node.js compared to the legacy system.

Aside from measuring success, the next steps involve a build-out of the entire environment — not just the apps that incrementally replace the existing solution, but the entire stack that supports that solution. The presumptive stack, ideal for modernizing operations, would likely include all of the following elements:

Node.js DevOps Stack

RUNTIME	**Node.js** Cross-platform JavaScript runtime environment for developing server-side Web applications.	**NodeSource N|Solid** Node.js runtime enhanced to address the needs of the enterprise. N|Solid adds a layer of security on top of core Node.js and provides greater visibility into resource usage.
OPERATIONS	**Containers** Packaging applications into a standardized unit allows teams to leverage immutable deployments and dynamic scaling based on demand.	
ORCHESTRATION	**Kubernetes** Cluster management platform to support the deployment, scaling, and operations of application containers across clusters of hosts.	

Taking That First Step

Digital transformation can be incredibly exciting, but also a little scary. And the biggest challenge to overcome may be figuring out how to begin. Your organization must first reach a consensus on where you are today and what you are hoping to achieve. Building out Node.js initiatives for the enterprise requires taking a broader view than organizations usually adopt when initiating a new development project. Successful Node.js deployments engage much more than just the team writing the code. Sustainable success comes from integrating people, process and technology — a perspective that incorporates the whole business.

Those who are taking the first steps towards a modern architecture have, in some ways, an easier path ahead than those whose businesses are inherently digital and who have been using Java up to this point. Just as Node.js enables much faster revision of existing environments, it enables much faster implementation of new environments. Additionally, organizations are increasingly updating their definition of what "modernizing" means, and what they expect to accomplish by pursuing modernization. Rather than seeing modernizing operations as something you do once, companies are beginning to see it as an ongoing evolutionary process.

NodeSource Assisted Transformations

Change is the new constant. Building a platform with Node.js and supporting tooling like the Node.js DevOps Stack (Node.js, Containers, Kubernetes) enable teams to thrive in the new digital landscape. At NodeSource, not only are we invested in supporting the open source Node.js project, we also provide product and support offerings to assist companies in their digital transformation.

> *"News Corp knows news. NodeSource knows Node. "*
>
> *- Jonathan Barnett of News Corp*

NodeSource N|Solid is a drop-in replacement for Node.js that provides greater security and performance insight into your Node.js applications. N|Solid helps you to identify modules with known vulnerabilities in your deployed applications and provides visualization of the resources your application is using. Node.js is a keystone of the digital transformation story, and NodeSource enables the adoption and success of Node.js within organizations of any size or sector.

Companies can also leverage N|Support, which provides access to the experts at NodeSource for solving issues that may arise during Node.js development. Architecture Evaluations are another offering where NodeSource can help guide your overall Node.js policies and strategy. NodeSource also provides world-class Node.js training, which can be customized to best serve your development team.

You can get more information on all of NodeSource’s offerings at https://nodesource.com.