Jordan Davidson

> Always make shovels

- Spanish Fork, Utah, United States
- thatoneemail@gmail.com 1
- 801-367-2950

GitHub LinkedIn About Me

About Me

I've spent my whole career, heck, probably whole life, trying to make processes better.

I remember when I was about 8 years old my mom had bought about 30 large packages of diapers on sale. While my mom was bringing them in I spent the entire time devising a way to string together a bunch of large party coolers so we could pull them into the house all at once like a big train. While I still have the same drive to keep improving my situation I am a lot more dicerning with what I spend my time on.

I love to read books about how to do things efficiently. My three favorite books right now are The Lean Startup by Eric Ries, OKRs Measure What Matters by John Doerr, and Grit: The Power of Passion and Perserverance by Angela Duckworth. These books show what's possible in an individual and in an organization with careful planning.

I have the most fun when I can enable developers to contribute more freely and, to steal a line from John Doerr, "Focus on what matters most" (Writing cool software). In this day and age there are many tools that can help or hurt this goal depending on how they are implemented.

Are you looking to make your development organization a Theme Park Experience? I want to help. Together we can make your company a great place to work.

Work History

Work Histories are split into summary, tech, and story for easy skimming.

Domo, Inc. - Software Engineer

Mar 2015 - Present

Summary

I tried to keep this list short. There were a lot of other things I did while at Domo, including stuff I actively maintained, but the list is just too long.

- Kubernetes configuration management platform.
- Cluster monitoring
- Contributions to haproxy-ingress to add required functionality
- Configuration validation system.
- Developer tools to create and destroy entire copies of Domo infrastructure to develop services locally.
- CLI tool used for interacting with the management platform.
- State capture to retain data between scale-downs.
- Auto nightly scale down for development environments to reduce cost.
- Build tools to auto package and templatize docker builds.
- kube-valet auto-scaling and "pack left"
- Performant custom Logging infrastructure with dynamic routing, filtering and transformation.
- Automated Cert renewal system.
- Initiative for getting Linux developer laptops approved for use at Domo.
- Caching proxy for docker registry.
- Okta compatible authentication bridge for Kubernetes

Tech Used

I know there is a lot in this list, but I have pared this down to the things I built things with or developed against on a regular basis.

Languages

!no-header!	!no-header!	!no-header!
Node.js	Typescript	Golang
Ruby	Javascript	Java
Python	Scala	BASH

Major Libraries

!no-header!	!no-header!	!no-header!
Spring	Express	Yargs
React	Inversify	Electron

Clouds

!no-header!	!no-header!	!no-header!
Google Cloud	Amazon Web Services	Microsoft Azure

Developer Tools

!no-header!	!no-header!	!no-header!
Linux	Visual Studio Code	tmux
K3s	vim	Jenkins
Gradle	gulp	Docker

Other Tools and Services

!no-header!	!no-header!	!no-header!
Kubernetes	Fluentd	ElasticSearch
Grafana	Kibana	Alert Manager
HAProxy	Nginx	Prometheus
Sumo Logic	Let's Encrypt	Okta
Gogs	Salt	Spring Cloud Config Server

Story

I got hired at Domo in the QA department. One of the developers was confused as to why I wanted to be in the QA department when I had so much experience with writing software. I told him that it's because I enjoy automating things and writing developer tools. It turns out that being in QA and doing what I have done is unusual in some places. QA at Domo ended up not being a great fit for me. I switched over to the Delivery Engineering team A.K.A. Voltron. There I designed and built an internal product that replaced developer environments previously run on a local VM with a LOT of swap space. The new product runs on Kubernetes and manages about 60 or so custom developer environments every day, spinning nodes up and down as resources are needed. It is currently slated to replace the production environments as well. I also facilitated many migrations within the company, and wrote about 15 or so micro-services used to support the Kubernetes infrastructure. I also built an auto updating Electron/Node.js based CLI application that among other things, tunneled a developer's laptop into their environment, authenticated kubectl commands, provided a GUI interface to the environment, and created and destroyed the "leased" environments.

Jive Communications - Developer of Awesome

Dec 2013 - Mar 2015

Summary

- WebSocket implementation of custom TCP messaging transport called JiveWire.
- Javascript port of JiveWire.
- Phone automation system using RaspberryPi boards to connect to SIP Phones to automate testing of the phones.
- Server to manage "checkouts" of test phones using bankers algorithm to prevent deadlocks in automated tests.
- Messaging transport to send commands to phones while "checked out".
- Software used to spin up dependencies of services for integration tests.
- Implementation of various angular bits for the new company platform.

Tech Used

!no-header!	!no-header!	!no-header!
Node.js	javascript	Java
Guice	Jetty	Maven
Jenkins	Angular 1	RaspberryPi
MacOS	Linux	WebSockets
BeagleBone	PostgreSQL	SIP Phones
Lombok	Intellij	Eclipse
Docker		

Story

When I started at Jive Communications I was hired to build an automated testing platform to make physical SIP phones interact with the development environment to ensure edge case test coverage. SIP phones rely on the SIP standard to communicate. However, often manufacturers will deviate from the standard to add intentional incompatibility or add competitive features. This system was devised to just use actual phones instead of guessing how they worked. A Simple test involved checking out two phones, making one dial the other, converting text to speech, playing the audio over the microphone of one phone while the other recorded audio going over the ear piece, converting speech to text and comparing the end result text to the text used at the beginning (whew). I ended up developing the system largely on my own due to shifting resources. In case you were wondering, yes "Developer of Awesome" was my official title. I put it on my loan application for the house I bought while employed there.

Novarad - Lead Automation Engineer

May 2012 - Dec 2013

Summary

- Automated tests, testing frameworks and reporting systems.
- Links between automated test results to Microsoft Test Manager.
- Software to run distributed tests across multiple windows machines.
- End to end HL7 testing framework used to automate tests that had been manual for years.
- Database integration test framework and tests used to ensure quality through a large migration from MSSql to Postgresql.
- Test harnesses for starting up databases and other services on laptops automatically.

Tech Used

!no-header!	!no-header!	!no-header!
Windows	Visual Studio	CodedUI
Team Foundation Server	MSSQL	PostgreSQL
C#	HL7	Javascript
IIS	Windows Server	Microsoft Test Manager

Story

At Novarad I joined as a QA Test Intern one semester into the Computer Science program at UVU. I had been learning C# and Novarad's platform was written mostly in C#. I got really tired of running the same tests over and over again manually. I enlisted to help with writing automated testing with CodedUI (Selenium for Windows programs and Internet Explorer).I proposed that we start taking screenshots of the most unchanging parts of the application to assert whether or not something had changed. Later when the results came in we could assess if those changes had been intended or not. I was way out of my depth, but I managed to pull it off. After a while the only other test engineer working on automation moved to development. I took over the automated testing program for the next year and added two more interns to my team before moving to Jive Communications.

Tech I love (And why I love it)

Kubernetes

Kubernetes has completely taken the development world over. Every cloud provider is tripping over themselves to create an offering even if it is just checking the box. This is not without reason. Kubernetes decouples your physical (or cloud) infrastructure details away from your application deployment. Because the interface to Kubernetes is just a CRUD REST interface, automating and extending Kubernetes is simple and straightforward.

Docker

Packaging your apps in Docker is one thing, but Docker let's you package your builds as well. Docker can make your workstation, ci/cd server, and your dev and production environments the same experience. Shipping your app for a quick test in dev can also be done right from your laptop without checking any code in.

Typescript

> Never put off until run time what can be done at compile time. - A. Glew

Typescript has the most advanced and flexible type system I have ever laid eyes on. I recently rediscovered just how amazing it is when I went back to writing java code. There are things you can express in typescript types that you just can't do in any other language. Adding types to Javascript with typescript makes it my favorite language to work with.

Node.js

Node.js Is an amazing backend platform. With just one thread to worry about it can perform complex async operations without having to deal with locks, or thread pools. Node.js is exceptional for creating stateless services that can be horizontally scaled. In ever evolving microservice-based architectures, horizontal scaling is easy and dynamic. The Node.js ecosystem is also amazing. Every library, strategy, and opinion is constantly challenged and improved. It can be a bit chaotic but I love it.

Linux

My favorite thing about Linux is the amount of choice it offers. It works well as a desktop for development, for an internet machine for your grandma, and best of all servers. Being able to run the same operating system on my laptop and my servers is amazing. Being able to tweak my desktop to work just how I want it (when time allows) makes me feel empowered as a developer.

Bash?

Ok, I know what you're thinking. Bash, really? I mean just look at that website. Is it 1990 again? Bash itself isn't the best language per se but the paradigm is so great for building plumbing especially infrastructure. In most languages/platforms, stringing together disparate technologies usually requires SDKs or other libraries. In Bash, stringing together disparate systems is idiomatic. Heck, even `[` is just a program on your `PATH`.

Terraform

I've recently discovered Terraform, and I love it. Any time you change a process from imperative to declarative, you transform Projects into Products. Terraform does a really good job of turning infrastructure tasks into declarative code, enabling GitOps in the cloud.

Prettier

> Consistency is almost always better than being right - Professor Welborn

Prettier takes the politics out of formatting. Just create a settings file, turn on auto format and you are done. You can even use prettier to auto format existing commits while retaining git blame. When you have a hyper consistent formatter your git diffs are all about the code and not about the formatting.

Mechanical Keyboards

If you can't handle me at my clackiest... You probably don't care, but I do. I love mechanical keyboards. I follow mechanical keyboard communities, I own a goofy split keyboard that glows rainbow, and I think $140 is pretty cheap for halfway decent keyboard.

Books I Love

The Lean Startup

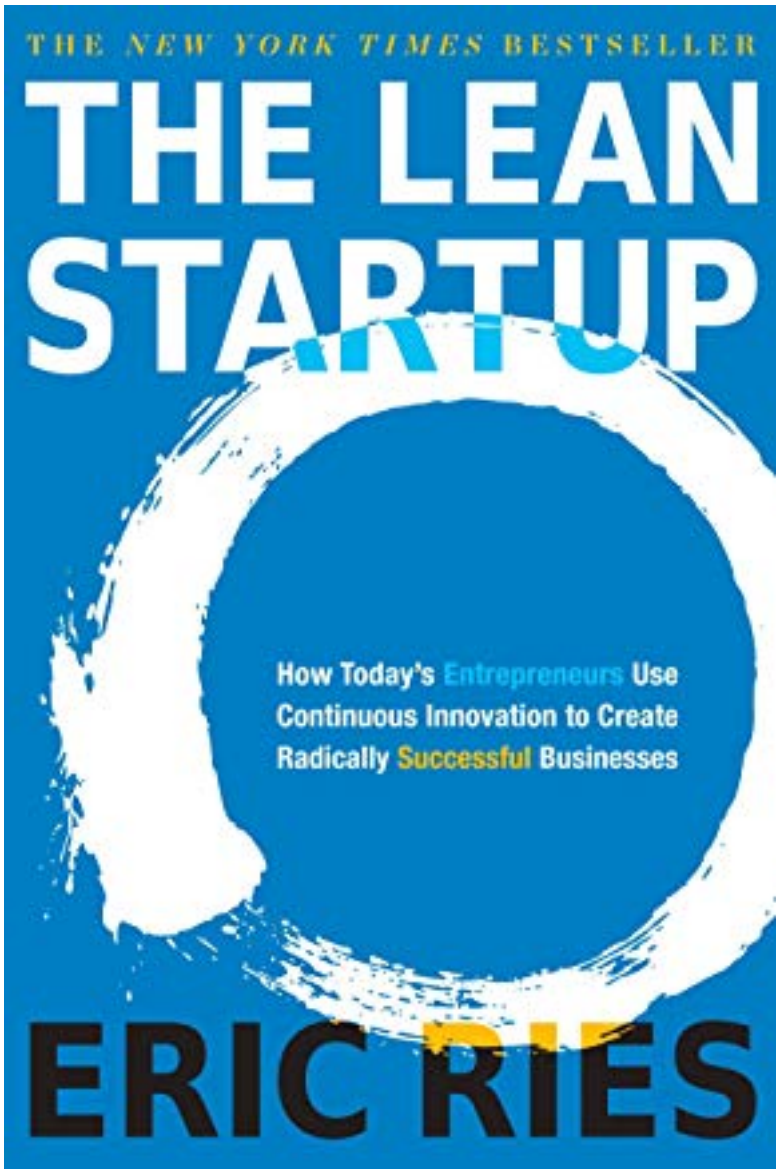

image taken from amazon.com

Eric Ries explains beautifully how to apply the scientific method, and the lean manufacturing methodology, to software development and design. Rather than

building what your customers say they want, focus on measuring customer behavior, and experiment to figure out what works.

OKRs Measure What Matters

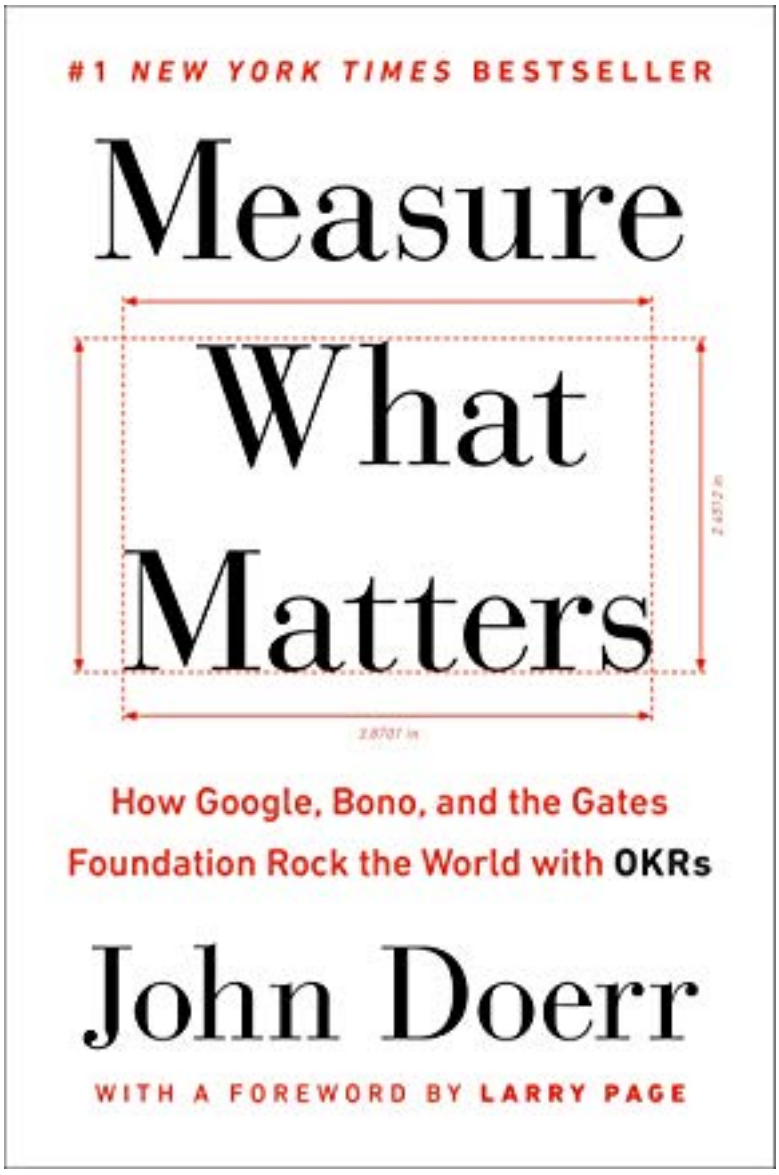

image taken from amazon.com

John Doerr recounts his experience with his mentors management system. OKRs (i.e. Objectives and Key Results) framework how to prioritize what is most important to be doing, help employees stretch for excellence, and inform leadership the capacity and direction of the company. With well applied OKRs there is less confusion and more company unity.

Grit: The Power of Passion and Perserverance

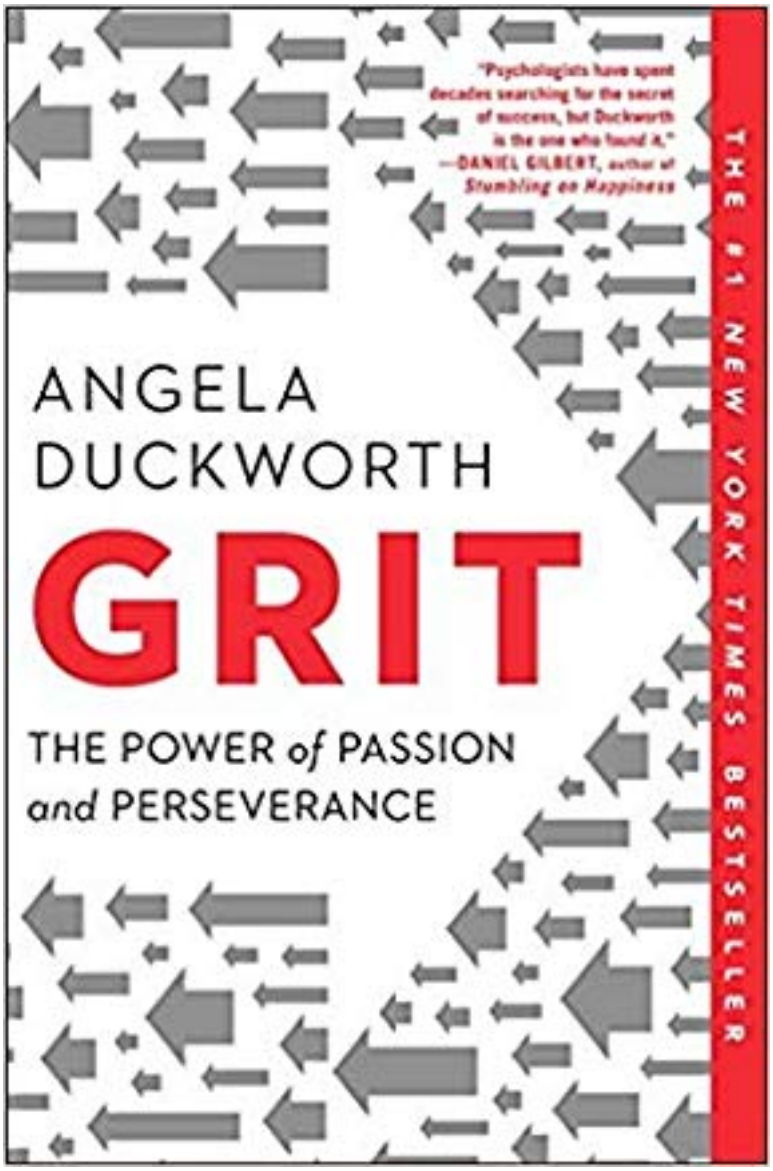

image taken from amazon.com

Angela Duckworth summarizes her extensive research into what makes people successful. She found that grit, the ability to stick with something to it's completion, is the number one determiner for successful people. She explains how you can become more gritty, and how to help those around you do the same. Top level goals are a main focus in her book and give direction on what you

should be gritty with and what you should give up on.

The California Gold Rush

In 1894, California experienced what we now call the "Gold Rush". The quote on my cover page refers to the difference in strategy between different enterprises and how they turned out. There are two main camps: the prospectors, and the companies that sold "shovels". "Shovels" really refers to the supplies, tools, etc. that enabled the prospectors to do their job. The companies that "made shovels" ended up making the most money, and had a more stable business model. Levi Strauss, who sold blue jeans to prospectors, was one of the biggest successes from the "Gold Rush".

What's your point?

I like enabling other people to write great software. It's not the most glamorous, but for me it's the most satisfying. When writing tools, building systems, or designing processes (making shovels), for people I work with I get extremely quick feedback (good or bad), and can act on that. I love making peoples lives easier, and I love programming, so I especially love making programmers lives easier.

Product VS Project

Working on a project is a lot like building a house. You can only build one at a time. Each time you build a house you have the potential to get better and faster, but you also have the opportunity to make mistakes. You can make the same mistakes over and over again. You can even make mistakes on things that you did perfectly the last time you built a house. The way to make your situation better is to make yourself more disciplined and focused but that can fail you if you are tired or distracted.

Creating a product is like building a factory. Every time you create a widget it comes out the same. If there is a problem with the machine all of the widgets will be created with a defect and it will be easier to figure out where the problem is. If you fix the problem, you fix all future widgets. The work you do on the machine has a multiplicative effect on the output.

Software development can be either one. When software development is treated like a project it becomes unstable and requires a lot of manual effort outside of writing the actual code. When software development is treated like a product the main focus can remain on features and code quality. (see also Theme Park Experience). A lot of this comes down to the tools surrounding your process. I

enjoy building tools, refining processes, and establishing cultures that facilitate a product mindset within software development organizations.

Theme Park Experience

When you are at a theme park everything is taken care of for you. You don't have to worry about the details of how the park is run. The only difficult part about being in a theme park is picking which ride to get on.

Having a theme park experience at a development organization means developers can focus on delivering features and not on repetitive tasks, red tape, or figuring why it's "broken in dev" but it "works on my machine". When developers are free to focus on solving problems, not only are they more efficient, but they are more satisfied with what they are doing and as a result: happier.