# Nik Ogura

### Principal Staff Engineer

*Platform - Tools - Security - Infrastructure*

*I make things - things that work- and by ‘work’ I mean work superlatively.*

### San Diego County, USA

#### *Aut viam inveniam, aut faciam.*

*(I will find a way, or I will make one)*

[*Reader’s Digest Version of this Resume*](https://github.com/nikogura/nikogura.github.io/blob/master/NikOguraResume.docx?raw=true)

# Table of Contents

[Interesting Things I’ve Done](#interesting-things-i've-done)

[Skills](#skills)

[Open Source Projects](#open-source-projects)

[Online Profiles](#online-profiles)

[Employment History](#employment-history)

# Interesting Things I’ve Done

[Orion’s SaaS and On-Premeses Kubernetes Systems](#orion-labs---san-francisco-ca)

[Scribd’s SIEM System](#scribds-siem-system)

[Scribd’s Managed Secrets System](#scribds-managed-secrets-system)

[Stitch Fix’s Algorithms Department’s Access and Identity System](#Xe9f496ea97e3008af73a0aa4184cc88b2f8ed85)

[Bent AWS’s Metadata System to Enable IAM Authentication on Laptops](#X0ccc14d66b4812033870135f93ca0fb87c4cab2)

[Invented a Self Updating Signed Binary Tool Distribution and Execution Framework](#X9b2f6858c1bd57e5f752cfbadada0aab57d5b14)

[Apple Pay’s Test Driven Cloud Based CI/CD Pipeline](#X230b5216f7f3e1d73394bf64a15e32026a3ec57)

[Made an Application Stack Prototyping and Orchestration Suite](#Xf0b87b5a7a1d560ca5719b1e83a85d9e3923708)

[Built Static Code Analysis Tools for Puppet Modules](#Xf9543268b6e368954a9a9c80e68ff3523a32a54)

[US Bank’s Encryption Key Management and Delivery System](#X191a60dd65ee2bf45c78c0b73312602e69ae6c8)

[Designed a Credit Card PAN Encryption and Tokenization System](#X7d59f1a275147d2368520bb7b3500382a6f1b2f)

[Made LAMP Stacks Galore](#made-lamp-stacks-galore)

[Built and Maintained US Bank’s Web Application Firewalls](#Xf9bf63395b9080397de41890cbac4836f9d54cb)

[Brought a Whole Business Line’s Tech Stack into PCI Compliance](#X96eb1ec03565335c57920fa027f22cc7dfb462b)

[More UI’s Than You Can Shake a Stick At](#more-uis-than-you-can-shake-a-stick-at)

#### Orion’s SaaS and On-Premeses Kubernetes Systems

The platform on which all of Orion’s technology stands. What we run for our SaaS customers is what we sell as an on-prem solution. We eat our own dogfood, and it’s delicious.

Picture a stand-alone, self-bootstrapping, one click Kubernetes based system that works in on-prem, cloud-prem, and even air-gapped installations. In addition to Orion’s PTT stack, the system sports a metrics and visibility stack, as well as its own auto-unsealing certificate authority powered by Hashicorp Vault.

The real power of the system is its UX. You enter a command, and it creates itself *ex nihilo*. Huge power, amazing complexity, yet it *just works*.

One customer evaluating the system described it as ‘flawless’. Another large customer called it “The easiest, highest quality software installation we have ever seen”.

*Components:* **Kubernetes**, **Go**, **ElasticSearch**, **Logstash**, **Kibana**, **Fluent-Bit**, **Hashicorp Vault**, **Prometheus**, **Grafana**, **AlertManager**.

#### Scribd’s SIEM System

Scribd’s world-wide footprint creates interesting challenges from a monitoring and abuse standpoint. Merely being able to see what’s going on is a challenge. There’s so much data coming in that ‘spinner disks’ can’t keep up with it and start smoking the moment you turn the system on. I had to write code that could receive, process, correlate, and consume information for processing. With it we discovered all sorts of interesting things- better insights into how our legitimate users were using the product, and also the bad actors and their botnets.

It’s all available in a self service fashion that allows anyone in the company to answer for themselves the question “What’s going on?”.

*Components:* **Go**, **ElasticSearch**, **Logstash**, **Kibana**, **ElasticBeats**, **Syslog**

#### Scribd’s Managed Secrets System

Imagine a YAML interface simplifying Hashicorp Vault.

Define your secrets- what they look like, and how to generate them. The system takes care of the rest.

A developer can define a secret, and who should access it, but not be able to know the value outside of a dev system. Any user gets the proper value for their environment.

Authenticate via LDAP, TLS Certificate, Kubernetes, IAM - it doesn’t matter. One binary tool magically does the right thing and ‘your secrets’ magically appear at your fingertips.

[Managed Secrets](https://github.com/nikogura/managed-secrets)

*Components:* **Hashicorp Vault**, **Go**

## Stitch Fix’s Algorithms Department’s Access and Identity System

It’s the means by which an entire department of Data Scientists and Engineers connected to every system, instance, and container in the stack. We wanted an an independent system that could funciton autonomously. The whole access and identity lifecycle- onboarding, offboarding, partner connections, and a light weight roaming profile for any user on any system in a dynamic auto scaling cloud environment.

The rest of the company used a 3rd party SSO offering. The 3rd party got breached and the SSO went down. We stayed up.

Here’s how you can do it too: [Access and Identity Made Easy](AccessAndIdentityMadeEasy.md)

## Bent AWS’s Metadata System to Enable IAM Authentication on Laptops

I worked out how to bend AWS’s metadata service so that we could leverage IAM Roles and STS credentials as if we were running in EC2, but actually do so with passwordless authentication on a laptop. The idea to even try came from others, but I’m the one who worked out how to make it happen.

Absolute black magic that *just works* and requires zero code alteration of Amazon SDK code and no monkey patching whatsoever.

Code works the same locally as it does in the cloud. No surprises. That’s what it’s all about.

Here’s how I did it: [IAM Beyond AWS](IAM-Beyond_AWS.md)

## Invented a Self-Updating Signed Binary Tool Distribution and Execution Framework

It’s used for distributing and running signed binaries on user laptops, in cloud instances and docker containers. It’s always up to date, works on and offline, and best of all it *just works*.

It builds, tests, publishes, and even updates itself *in flight* when there’s a new version available. The binaries are signed, so you can verify integrity, trust, etc. It works so automatically and magically that I had to make sure it couldn’t be easily made to do evil.

When I figure out how to get it to actually *write* it’s own new features and fix it’s own bugs, I’ll have invented SkyNet.

Here’s how I did it: [Self Updating Binary Tools](DBT.md)

Here’s a version you can use: [DBT- Dynamic Binary Toolkit](https://github.com/nikogura/dbt)

## Apple Pay’s Test-Driven Cloud-Based CI/CD Pipeline

Apple Pay and the systems behind activation of every Apple device in the world used it.

It worked pretty well, here and abroad. I don’t know how much of it they’re still using.

Hear about them going down during the China launch and being unable to deploy between 2015 and 2017? Me neither. Never happened.

The system was actually one of the best CI/CD systems I’ve had the pleasure of building. Philosophically it was the ideal setup. It dynamically created an entire virtual environment for each and every pull request. That environment lived until the PR was merged to master, when it was torn down and the resources returned to the general pool.

We did it by putting OpenStack on a bunch of unused hardware that was lying around still in the original boxes. Virtual machines, once created, were provisioned via Chef. The hardest part was getting the blades into a rack, since the datacenter was still under construction at the time.

It was so secure that Apple’s own security teams could not get into it until we stood down some of its protections. I was told that was a first.

## Made an Application Stack Prototyping and Orchestration Suite

I worked up a system whereby the entire deployment footprint of a group of applications can be described and manipulated in code. It was self healing and could detect and correct drift.

Applications, Load Balancers, Firewalls, you name it. Not only can it be managed ‘for real’ in the datacenter, but it can be virtualized in a private cloud or locally in docker up to the limit of the hardware.

This was before things like Docker Compose was really stable, and Kubernetes was just beginning to gain serious traction.

In our case, we used the real Puppet and Chef modules used to configure the apps in the data center to build the containers locally for local integration testing.

It’s not a model I’d suggest repeating. There were way too many moving parts. Building docker containers with Chef Recipies and Puppet Modules is kind of icky to even think about, but it worked.

In this case we needed to prototype the ‘real’ applications locally, as the real running instances were all precious snowflakes that nobody could reproduce. Taking what was running ‘for real’ and making it build us ephemeral local models for testing was the way to go.

Here’s a sanitized and generalized example of how I did it: <https://github.com/nikogura/boxpile> Nowadays, I’d just use ‘Docker Compose’.

## Built Static Code Analysis Tools for Puppet Modules

There were no tools to do SCA of Puppet modules for GRC (Governance, Risk- Management, and Compliance). So I made some.

The idea was to state categorically pre-merge and pre-deployment that a change to the code would not bring the system out of compliance with regulatory and security baselines, et al.

I got to delve into the world of Lexers, Parsers, Tokens, and language processing- all that wild stuff. It was a blast.

## US Bank’s Encryption Key Management and Delivery System

The company bought a PKI, and it sucked. The parts didn’t talk to each other. It couldn’t deliver the keys. Keys were being left on shared drives and pasted into group emails. *Ouch.*

So I worked up a system to tie together components from multiple sources together and automate deployments. It allowed one button delivery of encryption keys to multiple operating systems across multiple network tiers.

It had a friendly user interface allowed non-technical users in multiple countries to create and deploy SSL Certificates in a secure manner. The application saved the company over $1M based on vendor bids for similar systems.

What’s even slicker is I taught myself Java to do it. From what I hear, they’re still using it today. I don’t know whether I should be proud of that or scared- it was, after all my first Java app. I’m sure they’ve polished off it’s rough edges by now. I hope.

### Designed a Credit Card PAN Encryption and Tokenization System

It was scalable. It was highly available. It was downright nasty to try to exploit. It encrypted and masked credit card numbers for a Merchant Acquiring systems (Credit Card Authorization and Settlement).

I also built and maintained the secure management tools for same. We had to hide the numbers, sure, but this was actual payment processing inside the financial sector. *Someone*- the right someone, and only the right someone internally had to be able to see the numbers, but no one else.

## Made LAMP Stacks Galore

My first one let us use the latest, most up to date versions of Apache, OpenSSL and Perl to run secure apps on servers that were at least 5 years out of date. 5 is a lot in ‘computer years’.

The stack was modern, but ran on an ancient systems. Their priesthood lovingly maintained them as the precious works of dystopian art they were. Meanwhile *we* did business in *this* century. To each, their own.

I’ve done the same for Python, Gunicorn, Django and Nginx.

## Built and Maintained US Bank’s Web Application Firewalls

I’ve designed, implemented, and maintained Web Application Firewalls for cross platform applications, some directly handling credit card PAN data.

Actually there were two of us. The other was my boss at the time Eric Malone. We handled the WAF’s for the entirety of US Bank at the time. It wasn’t our role, but we did such a good job of managing our own, we kept being asked to bring other apps and teams onto our firewalls. The WAF policy for all of US Bank at one time boiled down to ‘Nik says so.’. It worked, because I wasn’t afraid to say ‘no’ when it needed to be said, and Eric backed me up with the upper levels of management.

We proved our compliance with PCI DSS and the OWASP Top Ten. You can install firewalls off the shelf with the push of a button. The trick is knowing enough about what is going on to tune the rulesets and educate the app developers to understand why the ‘easy’ and ‘simple’ construction they like to use is actually opening the system to attack.

Who can do that? This guy.

## Brought a Whole Business Line’s Tech Stack into PCI Compliance

I participated in creating and executing a plan to bring a multi- million dollar business line centered around credit card processing systems from zero to PCI 2.0 compliant in < 12 months. Can do it again too.

## More UI’s Than You Can Shake A Stick At

Lets be clear here, I’m no right brained web designer. I pride myself on functional, not pretty. Take a gander at any picture of my ugly mug and you’ll see why.

Dynamic Javascript? check. Ajax? sure. Wacky cool Javascript Frameworks? I’ve used ’em, and even written them.

GUI programming? Done it. My strengths run to the working rather than the fancy, but I can make you pictures of buttons to push if thats your goal.

# Skills

[Communications and Teaching](#communications)

[Computer Languages (To Date)](#computer-languages-(to-date))

[Networks and Traffic Management](#networks-and-traffic-management)

[Quality, Security, and Testing](#quality-security-and-testing)

[Cloud Goodness](#cloud-goodness)

[Containers ’n Stuff](#containers-'n-stuff)

[Making Things Work](#making-things-work)

## Communications

I’m primarily a teacher, and an evangelist. Mom wanted me to become a minister. I would have been good at the ‘fire and brimstone’ parts, but I can’t pull off the high hair.

With over 30 years experience in bringing the light of knowledge to the dark corners of the intellectual realm. If I know something, I can help you know it too. If you want to know it that is. No power in the ’verse can make you want to learn something you don’t. You can lead a person to knowledge, but you *cannot* make them think.

I’ve written curriculum for courses in Test Driven Development, Application Security, Network Security, and some light Penetration Testing. I’ve taught the courses too.

I’ve been a coach for public speaking and performance, was an instructor for leadership, taught both elementary and secondary classes, and was even a wildlife educator at the San Antonio Zoo in a past life.

Things I can do are great, but it’s really all about what I can help others achieve. See, **I** don’t scale. There are only so many hours in the day and my fingers move only so quickly on any given keyboard. But if I tell two people, and they tell two people, and so on and so forth, we have a whole revolution on our hands.

One of my kung fu teachers used to demonstrate the most amazing magical abilities and leave the audience dumbstruck. Then he’d ask “Pretty cool, huh?”. There was never any hesitation. “Yes! Amazing!”

Then he’d say “No. It’s only cool if **you** can do it.”. Need I say more?

## Computer Languages (To Date)

Computer languages are like brushes to a painter. Which is my favorite? Favorite for what? They’re all wonderful in some way, and they’re all miserable in some way. You try to use the right tool for the right job and not get dinged too much by the rough edges.

So far I’ve been deep into: **Java**, **Ruby**, **Python**, **Groovy**, **JavaScript**, **C**, **Perl**, **Go**, and **Bash**.

My skills in all of the above rise and fall as each gets used and then left to erode while I blow the dust off the next. I rarely spend enough time in any single language to really feel like a virtuoso anymore.

Once upon a time I read and wrote Perl better than I read and wrote English, but those days of narrow focus are long behind me.

If you want to whiteboard some code, I can prove to you I know how to write it. It might come out as psudeo-code or *JavaGoPerlThon*. That being said, it doesn’t take long at all for me to get deep again into the language of the day.

## Networks and Traffic Management

I can claim broad and occasionally deep knowledge of how computers talk to each other.

I grok the layers, and get the various things that happen at each one, though I confuse the 7 layer OSI model with the 4 layer TCP/IP model and have been known to forget who calls which layer what. What’s in a name anyway?

I’ve written routers and firewalls from scratch, know how packets travel around, built VPN’s, and traced network traffic to reveal poisoned ARP Caches. That was fun.

I’ve run set up and maintained BIND, and can both *cause* and *resolve* DNS issues across Local and Wide Area Networks.

IPTables and PF (Packet Filter) do my bidding. The packets go where *I* tell them to go.

I’ve built IPSec tunnels, and have unwound the Linux networking stack with it’s network namespaces to the point where I could run OpenStack *inside* of OpenStack and have it all mostly work. Why? Long story. I’m still shaking from the experience. I did however come away with a one-file installer for a private OpenStack cloud that I keep on my key chain. Want an instant OpenStack?. [One Script OpenStack Installer](OpenStackLibertyInstaller.md)

I get load balancers and the routing of traffic. I’ve used Open Source ones such as Apache, Nginx and HAProxy, and some of the ones with price tags such as F5 BigIP, Netscaler, AWS ELB. I can cross and uncross the streams at will.

Here’s an example of doing some really nasty and wonderful things to a laptop’s networking stack: [IAM Beyond AWS](IAM-Beyond_AWS.md)

## Quality, Security, and Testing

I’m a big believer in Testing, and Test Driven Development. Actually that’s not quite right. I’m actually a big fan of **Integrity**, as a person, as a coder, and as a citizen of the universe.

What is Integrity? Quite simply it’s knowing what something *is* and what it *is not*, and being able to continuously demonstrate that for any interested parties. This is the essence of QA, Security, and Trust.

We can’t begin to talk about whether something is good or bad, secure or insecure until we establish exactly what we’re talking about. That’s where code signing and secure, repeatable, transparent build tooling comes in. It is exactly what we think it is and contains nothing we would find surprising. Surprises are fun if they involve cake and paper hats. Software? not so much.

Once we can establish what it is, we can start to describe it’s behavior, and the interface contract between the thing and it’s users. That’s really what **Testing** is: proving to ourselves and any other interested parties that it does what we expect.

Moreover, you should never take my word *(or anyone else’s)* for it. Perform the experiment yourself and observe the result. That’s what science is. They do call it *Computer Science* for a reason.

Some test supporting work:

* [gomason](https://github.com/nikogura/gomason) A tool for doing clean-room CI testing locally.
* [go-postgres-testdb](https://github.com/stitchfix/go-postgres-testdb) A library for managing ephemeral test databases.
* [python-ldap-test](https://github.com/zoldar/python-ldap-test) A testing tool Python implementing an ephemeral in-memory LDAP server
* [Selenium4j](https://github.com/nextinterfaces/selenium4j) A Java Library for translating HTML format Selenium tests into JUnit4 at runtime.

Are tests enough? Depends on the tests, and the circumstances. Write crap tests and get smelly results. Garbage in, garbage out. No shiny tool can change that.

If we can continuously prove that our code does what it’s supposed to do and that we have full provenance of it’s history and lifecycle, then **Security** becomes a snap and auditors aren’t so scary anymore. They’re human too, they just like you to *prove* things- as they should.

It’s actually pretty easy if you develop the discipline. It just looks like a bunch of meaningless work in the beginning until you build the muscles. Once you’ve built the skills and the habits, you’re just doing your thing, but the bad guys just bounce off like so many bugs on the windshield.

Proof is also not a one time thing. To be really meaningful, it must be continuous just like ‘integration’, ‘delivery’, and ‘feedback’. With any sort of one-off, we can only say ‘it worked last time’. Well, great, it used to work, but things change so quickly these days that even if your code didn’t change, the rest of the world has hardly been standing still.

It might not ‘work’ any more. It also might not be ‘secure’ any longer. When that happens, you *might* still pass the audit, but your ‘Security’ has now become ‘Security Theater’.

‘Running to Stand Still’ isn’t just an old U2 song. *(Yes, I’m that old.)* Lace up those shoes and stretch. We’ve got some running to do.

If all you want to do is pass audits. Don’t call me. I’m just going to tick you off. If you want to both *pass audits* and *be secure*, let’s talk.

[Some more musings on TDD](TDD.md)

## Cloud Goodness

I’ve flown through (wandered around in?) the big clouds such as OpenStack, VmWare, and AWS without hitting the rocks. Am I a master? Hardly. Anyone who claims to be one might just be pulling your leg.

Are they cool? Hells yeah! Do I like them? I do indeed. Would I like to play in them some more? When can I start?

What do you want to do? Make stuff in clouds? We can do that. Wanna make clouds themselves? Got you covered there too.

What kind of cloud do you want to whip up? I’ll make you one that’ll register as a Category 5, but without the whole ‘flooding’ and ‘backed up sewage’ parts. Eew.

In fact, here’s a little gem I keep on my keychain. Want to create a full Open Stack Liberty cloud from nothing with a single script? [One Script OpenStack Installer](OpenStackLibertyInstaller.md)

Granted, that’s a little dated. Open Stack Liberty is pretty old school at this point. It will get you an idea of how I like to roll though. Enjoy.

*(Wait, you keep an OpenStack installer on your keychain? Yup. Because I can.)*

Nowadays everybody wants to talk about Kubernetes. I have you covered. I’ve built production ready K8S clusters. Several of them in fact. I’ve also built the systems that give people self-service access to them, and the ability to deploy code to them with the click of a button.

Familiar with the Kubernetes source code? Check. Blue/Green deployments? Ayup. Not just blue/green either. As many colors as you need- think cached precompiled javascript assets in a world-wide CDN stretching back in time. How many? How long does your cache live? Not only do you have to deploy them, you have to clean ’em up when you’re satisfied the pods are no longer in use.

## Containers ’n Stuff

Been there, done that. I wrote something funky to do local container orchestration for Apple. This was before Docker Compose was really stable. Now I recommend we all just Docker’s version, cos it’s open source and awesome.

I’ve written tools to build container’s locally, and deploy them to the cloud. Piece of cake. You can rapidly prototype locally, and deploy in a snap. Isn’t that what this is all about?

Here’s an example. <https://github.com/nikogura/boxpile> That’s basically ‘docker compose’ written in java, and using Chef to create the containers. Dear gods why would I attempt such a thing? Well, we couldn’t get buy in on replacing *everything* at once with Docker, so I had to come up with a way to demonstrate that doing X ‘in docker’ would be ‘just as good’. So I did.

Full end to end integration testing of the ‘real code’, configured the way it would have been ‘for real’ in the datacenter using the Puppet Modules and later Chef Recipes that were in use ‘for real’. Worked too.

Lately I have worked for a couple of shops that are strongly biased towards ‘built here’ solutions. On the one hand, this is cool, because it allowed me to get extremely deep into the wiring of this technology. On the other hand it’s annoying because there are so many perfectly good wheels out there, freely available. Some of them I’ve personally reinvented multiple times. All that work could have been applied to an open source solution that is enjoyed (and maintained) by many.

If we’d made these projects open source from the start or joined one in progress, not only would they probably be further along now, but the original author (me) would likely still be engaged in supporting them. Headcount and available time/energy/focus for any team is always going to be limited. When it comes to help, sometimes you gotta take ‘yes’ for an answer.

## Making Things Work

I think of a hacker as someone who can pop the hood and fiddle around inside- make it go, stop, or do something unexpectedly wonderful. Someone who is entirely unintimidated by not knowing something, and who’s eminently comfortable digging in and figuring it out.

Some of those sorts of people use that knowledge to bring ruin on others, and they have sullied that once noble name.

I still think of myself as a ‘hacker’, but in the sense I just described. I’ve had to learn more than a few of the tools the baddies use to do their nastiness, but I prefer to play defense. I’m the Safety, the Watcher on the Wall, the big scary bouncer. You’re not messing around here on *my* playground. Not on *my* watch.

Mainly I got into this field because I had a business to run, and knew I couldn’t do it all by hand. What I realized is, if you have a job that merely requires precision, speed, accuracy and repeatability, it’s a job for a machine, not a person.

The idea, to me anyway, is to get the machine to do what the machine does best, and free up the humans to do what only humans can do- the things that require imagination, judgement, artistry. Those are the fun jobs anyway. Who wants to be a robot? Well, ok, maybe a T-101 would be cool. Hasta la vista, baby.

I groove on making things that help people do things that would otherwise suck. Secure handling of encryption keys isn’t rocket science, it’s actually quite boring, but you have to do it right. Every time. All the time.

It doesn’t take much thought, but it does take great attention to detail. What’s more, one slip up can have huge repercussions. So I made the machine do it. Machines don’t skip steps and don’t get bored. Machines don’t get distracted. They just do what they’re told. If only I could get my kids to do the same.

What do you need to do? What are the pain points? I reckon that, with a little thought, a little sweat, we can find a way to make it not painful at all. It might even be fun.

If I cannot find a way, I will make one.

# Open Source Projects

* [dbt](https://github.com/nikogura/dbt) “Dynamic Binary Toolkit” A framework for authoring and using self-updating signed binaries. Listed in [awesome-go](https://github.com/avelino/awesome-go)
* [gomason](https://github.com/nikogura/gomason) A tool for doing clean-room CI testing locally. Listed in [awesome-go](https://github.com/avelino/awesome-go)
* [managed-secrets](https://github.com/nikogura/managed-secrets) A YAML interface simplifying [Hashicorp Vault](https://www.vaultproject.io/)
* [go-postgres-testdb](https://github.com/stitchfix/go-postgres-testdb) A library for managing ephemeral test databases.
* [python-ldap-test](https://github.com/zoldar/python-ldap-test) A testing tool Python implementing an ephemeral in-memory LDAP server
* [CGI::Lazy](http://search.cpan.org/~vayde/CGI-Lazy-1.10/lib/CGI/Lazy.pm) A Perl Web Development Framework.
* [Selenium4j](https://github.com/nextinterfaces/selenium4j) A Java Library for translating HTML format Selenium tests into JUnit4 at runtime.

# Online Profiles

[Home Page](http://nikogura.com)

[Code Repos](https://github.com/nikogura)

[LinkedIn](https://www.linkedin.com/in/nikogura/)

# Employment History

[Orion Labs - San Francisco, CA](#orion-labs---san-francisco,-ca)

[Scribd - San Francisco, CA](#scribd---san-francisco,-ca)

[Stitch Fix - San Francisco, CA](#stitch-fix---san-francisco,-ca)

[Apple - Cupertino, CA](#apple---cupertino,-ca)

[Data Recognition Corporation - Maple Grove, MN](#X4807bd6f415f812050ee53ee98cf807f705a1b3)

[Wells Fargo - Minneapolis, MN](#wells-fargo---maple-grove,-mn)

[US Bank - Minneapolis, MN](#us-bank---minneapolis,-mn)

[Plain Black - Madison, WI](#plain-black---madison,-wi)

[Hessian & McKasy - Minneapolis, MN](#hessian-&-mckasy---minneapolis,-mn)

[United Martial Arts - Plymouth, MN](#united-martial-arts---plymouth,-mn)

## Orion Labs - San Francisco, CA

2020 - Present *Principal Staff Engineer*

I took a legacy EC2 autoscaling application stack and re-architected it as a stand-alone, self-bootstrapping, one click Kubernetes based system that works in on-prem, cloud-prem, and even air-gapped installations. In addition to Orion’s PTT stack, the system sports its own auto-unsealing certificate authority powered by Hashicorp Vault.

While doing that, we replaced an expensive Splunk based monitoring/metrics system with a totally modern, best in class, and most importantly free stack based on Prometheus, Grafana, and Alertmanager. What’s more, since it’s based on open source technology, our monitoring/metrics stack is able to be bundled into our on-premeses product as a value add for our customers.

I’m also responsible for all technical training within the company. To date I’ve designed curriculum and taught internal classes on topics such as golang, elasticsearch, kubernetes, dns, IP networks, TLS/SSl, and both symmetric and asymmetric encryption.

## Scribd - San Francisco, CA

2018 - 2020 *Dev/SecOps Engineering Lead*

I created one-click self service deployment tooling to bare-metal hosts and Kubernetes clusters. Heck, I even created a series of Kubernetes clusters myself, ex nihilo, and lead the effort to use them in anger.

The company’s entire onboarding and access system, both to our network and our K8S clusters came out of my fertile mind and busy fingers, as did our internal PKI- with a little help from Hashicorp Vault and a ton of golang magic.

I designed and build a system of ‘Managed Secrets’ so that we could generate, rotate, and well, ‘manage’ secrets across the enterprise- in AWS and in a bare metal datacenter. An app getting the right secret is important, but you also need to know who has access to what, when to rotate, et al.

I tamed the ELK stack, and wrote event correlation tools to take incoming request data from Fastly’s WAF and make it available to detect and counter bad actors all over the world. This system ingests hundreds of Gb of information daily that flows in so quickly that it melts old fashioned spinner disks.

## Stitch Fix - San Francisco, CA

2017 *Data Platform Engineer Algorithms and Analytics Department*

Here’s where I made the Access and Identity system described above, and where I made the signed binary tool framework. It’s where I learned AWS, Golang, and really started grokking the internals of the SSH protocol.

This is where I worked out how to make self-updating tools.

## Apple - Cupertino, CA

2015 - 2017 *Senior DevOps Engineer, iOS Systems (Apple Pay)*

I’m one of two guys who Designed and built a dynamic test driven CI/CD pipeline for Apple Pay, Apple Sim, and every Apple device in the world.

I implemented a private OpenStack cloud for testing and verification of applications. From scratch. They handed me a pile of HP DL360’s and ‘do something with these’. So I did.

I authored a system whereby the entire deployment footprint of a group of applications can be described and manipulated in code.

I helped guide iOS Systems move from SVN to Git.

## Data Recognition Corporation - Minneapolis, MN

2014 - 2015 *Principal DevOps Engineer*

I designed an auto scaling Continuous Delivery environment for educational testing.

While there I shepherded multiple applications from proprietary systems to fully Open Source platforms. Goodbye Microsoft, hello Linux.

I designed and taught internal training curriculum for the technology, disciplines, and cultural concepts that come under the heading of DevOps.

I wasn’t there long. I would have happily stayed, but Apple called and sunny California beckoned. Can you blame me?

Ice in a glass or on a distant mountain is as close as I ever want to be again. *Brr.*

## Wells Fargo - Maple Grove, MN

2014 *Sr. Software Engineer*

I was a DevOps Consultant for Development, Testing, Building and Delivery of Applications and Middleware.

My main job was that of Module Developer for Continuous Integration/ Continuous Delivery of multiple applications across multiple technologies and multiple operating systems. They had multiple OpenStack clouds. We built the Puppet modules to deploy into it.

My thing was, and still is, automation of anything and everything- anything that a machine can do faster and more reliably, freeing the humans to do what only humans can do.

This is where I designed and built language applications to parse Puppet source code to do SCA for GRC.

## US Bank - Minneapolis, MN

2007 - 2014 *Applications Systems Administrator Sr.*

Mainly I did what I call ‘Specialty Application Development’. That is projects too sensitive or specialized for a general development team, or things that were deemed ‘impossible’. I didn’t know they were impossible, so I just did ’em.

This is where I built the encryption key delivery system, and my first LAMP stacks. This is where I did most of my credit card related goodness, where I learned about encryption, PCI, that stuff.

I was basically a ‘security consultant’ for an Application Architecture team. I worked with Arch and Development teams to evaluate threats, explain the scan findings, and show the folks how the vulnerabilities work and how to fix them.

This is where I really delved into load balancing and monitoring. I also presented internal courses/talks to business and engineering groups. I trained teams in automated testing tools and techniques.

Blah blah blah. It was a bank. They were serious about protecting people’s money *(I still use them, in fact.)* All that fun security/encryption/spooky paranoia type stuff listed above? I started doing it here.

## Plain Black - Madison, WI

2006 - 2007 *Application Developer*

I provided online troubleshooting for customers, and performed core development on the WebGUI CMS.

We wrote *and maintained* a 200k+ line ModPerl CMS. In Perl. It’s possible. You just have to have discipline.

The Perl compiler doesn’t force any rules on you, so you have to make some for yourselves, and stick to them. We did.

## Universal Talkware - Minneapolis, MN

2000 *NOC Administer*

I handled internal tools development, built the NOC, and even supported the physical plant. Dumpster diving for scrap metal for use in patching the server racks was a regular passtime.

I even had the interesting experience of programming and maintaining a PBX Exchange. Think green phosphor screens and keyboards you had to lean into to get a key to press. A PBX handles the routing of Publically Switched Telephone Networks. Ma Bell dropped the calls off at our door, and my PBX routed the signals to the analog phones. Old School. Seems crazy now, but it was a good experience.

## Hessian & McKasy - Minneapolis, MN

1999 - 2000 *IT Administrator*

I started out as the help desk, and ended up as the head of IT for a 40 seat Law Firm. That was over the Y2K change over, which, since we did our due diligence, was a total non-event.

## United Martial Arts - Plymouth, MN

1998 - 2007 *President, CEO, and Head Instructor (also Lead Programmer, Receptionist, and Executive Janitor)*

I was responsible for day to day operations of the martial arts studio, including management, financial planning, personnel- you name it, I did it.

I taught classes in Exercise, Wellness, Leadership, and the Martial Arts in the studio as well as for corporations and in the community.

Over the course of 9 years, I designed, built and maintained a custom studio management desktop application that handled enrollment, financials, lesson plans, scheduling, video and print library management, and curriculum.

Coding was cheap entertainment. Kick and punch doesn’t pay much. Somewhere in there I guess I got good at it, because it turned into a whole new career- one that involves considerably less head trauma.