Script

Slide 2

Hi, I'm Kenny Johnston and I'm super excited to be here. Giving a talk at Monitorama is definitely on my bucket list. I'm going to admit I'm a little bit nervous, but this topic is one I'm really passionate about and one that has surfaced in a many different aspect of my life. We're going to talk about the tension between Artisanship and Automation and how that tension relates to the standard devops tool of runbooks. Credit for the principles go to my lovely partner, the amazing teams I've work with, a book called Drive by Daniel Pink and countless previous Monitorama talks I've watched.

Slide 3

A little about me, I live in Kansas City - I'm an engineer by education, a product manager by trade and an avid learner in spirit. I had a first career working in political campaigns, an experience that taught me about what motivates people and then I had a second-career working in and around open-source technology which taught me the value of collaboration and people centric processes. I now work on the Product Management team at GitLab where our ambitious goal is to bring SRE style DevOps to everyone.

Before I start - I'd like to point out that what I'm going to describe is NOT how we operate at GitLab - parts of it ARE, but not all. What I'm going to describe is my personal proposal to make getting started in the continuous feedback loop for operations easier for DevOps teams.

Let's get started!

Slide 4

So - were here in Portland Oregon, I'm giving a talk at an observability conference that attracts "artisans within [their] team." What are some of the other names for Portland Oregon?
Rose City
Bridgetown
P-Town
Stumptown
PDX
Beervana?

Well whenever I think of Portland I think of "artisanal"- almost everyone I know from Portland has some sort of artisanal hobby. I don't think I'd be mistaken for calling Portland, Artisan-ville.

Slide 5

Artisan-ville - Portland is a city that supports it's artisans. Infact supporting small local businesses is a time honored tradition here. Lot's would argue that it is this support which drives Portland's famous quirkiness. In today's economy beset by automation and commoditization the drivers of local growth will in-fact be artisans. As the saying goes - don't mock the pickle maker. Portland isn't just local, it was local before anyone else. They are so far ahead of the rest of us - they are growing swiss chard at city hall. Seriously, that's true.

Slide 6

Beyond swiss chard and pickles, here are some of the artisanal things you can pick up in Portland. Beer, cocktails, costumes, jewelry… The gentleman pictured here is from a company called Portland Razor Co, the creator of hand crafted straight razors. Think about that, straight razors - the ultimate commodity - I mean you can get eight of them in a single razor these days. I find that amazing. Well that and I assumed Portlanders never shaved!

Slide 7

Artisanship is officially defined as "a worker in a skilled trade, especially one that involves making things by hand." Artisans are skilled and make things by hand. In Japanese the word for artisan is shokunin (show-ku-nin). As part of mastering their trade, shokunin also honor their tools. On New Years they honor their tools specifically to show their gratitude for performing their task.

Slide 8

There is a type of mindset that artisanship exemplifies. Is anyone here familiar with the book Drive by Daniel Pink? For those familiar, you'll recognize the artisan mindset as on centered around Mastery. For those unfamiliar Daniel Pink identifies THAT right brain activities, those that require expansive thinking - respond poorly to extrinsic motivators. What exactly does tha tmean? The example commonly used is called the candle expirement, and it showed that when asked to perform a task which required even rudimentary cognitive skills participants perform worse when given monetary incentives. That's right - on average, we perform non-menial tasks WORSE when given external incentives. Daniel Pink's book Drive goes on to highlight three critical components of intrinsic motivation for these non-mechanical tasks - Autonomy, Mastery and Purpose.That's what makes up the motivation of an artisan. Let's keep that in mind when it comes time to think about what tasks we should ask our teams to perform.

Slide 9

So - time to pivot - what's the opposite of artisanship? You guessed it - automation.

Slide 10

Automation - not the creation of automation - but automation itself is by its nature contractive. It requires no skill, no mastery. Despite similar spellings automation is decidedly NOT autonomous . Frankly, while often sophisticated - the means, the tools we use to perform automation all tend to become commodities. Of course in our industry all software becomes a commodity - but even in other industries the means of automation spur efficiency improvements (think about electricity, or the assembly line) but they quickly become standards, they quickly become commodities.

Slide 11

So, how would an artisan respond to automation. From a motivation perspective automation is mindless and repetitive. Remember mastery, autonomy and purpose? Automation requires no mastery, it provides no autonomy. Even those things that are highly purposeful, like pushing a very important button, are demotivating to those accustomed to the other extrinsic rewards of mastery and autonomy.

Slide 12

Whose most likely to react this way to automation. I honestly wonder if it is all of us but I'll give a few examples. Architects - architects have an artistic, almost artisanal bent to their work. Mechanics aren't fans of automation. Anyone here work on cars? For most people who do - the fun is in discovery. If you gave every mechanic a diagnostic book that exactly outlined how to fix every issue they'd probably throw it in the trash. I know my favorite mechanics, Click and Clack would. Doctors - doctors are a whole other story. I mentioned this talk was inspired by my spouse. Here's why - she works in administration at a hospital network and is tasked with reviewing clinical best practices (based on the most up-to-date science) with doctors. Doctors love discussing it, they love defining best practices - but their conformance to these best-practices - even when the best practices are the ones their peers agreed to - lags. Sound like anyone you know? This mentality is abound on DevOps teams. I once had team member state - "I'm not going to use those automation scripts, these are complex problems that require sophisticated skills. I'm an artisanal mechanic, I'll figure out the issue myself." Turns out even IT - in our hopefully deterministic world have a need for artisanal expression.

Slide 13

So in DevOps what types of automation are there. There are self-healing applications, their are monitoring alerts, their are standard diagnostic scripts, their are cleanup cron jobs, there is process documentation and there is … of course… runbooks.

Slide 14

Runbooks have been around a while. Typically relegated to traditional operational roles, they are often derided in modern DevOps organizations focused on pure automation. They are by their nature known prescriptive best practices for common diagnostic and remediation tasks. Anyone who has used them knows they are often useful but frequently insufficient. In sophisticated environments they are source-controlled and increasingly they are interactive - allowing notebook style retrieval of diagnostic info and script execution inline with written documentation.

Slide 15

The background of this slide is a statue in Dusseldorf. It was created in reaction to a late 60's student movement against the conservative West German government. You can tell by the look on the faces of the two men, it's meant to showcase the lack of understanding - the kind of talking past each other - between the two sides in a dispute.

I'm genuinely interest - who here is an artisan, who takes pride in the hand-craftedness of their capabilities? …. And who here is an automator - coming up with creative solutions to automate all tasks? ... Who is somewhere in between?

Interesting, let's talk about that conflict.

Slide 16

Side one - So we'll call the artisans the automation burners. As this part of the new testament staters, as depicated in this famous painting, - "Those who had practiced magic arts - brought their books together to burn." Burn the runbooks - leave us to our mastery!

Slide 17

Side two - But wait a minute. My OTHER bible - Google's SRE Manual - clearly states that the real black magic is everything that isn't automation. I should be out to automate all the things! I should be automating myself out of a job! Anything left to humans should be immediately crushed!

Now - I'll point out there is a small, overlooked paragraph in the chapter about automation which states:

"[In the above we've] stated a maximalist view of our position… in general, we have CHOSEN to create platforms[automation] where we could, or to position ourselves so that we COULD create platforms over-time."

Could create platforms overtime… Interesting. It would seem the gods at Google have offered some recognition of a middle ground. Have we pushed to quickly in our dogmatic automation approach? I think this unexplored middle-ground is crucial for our industry. The lack of depth here is actually what is inhibiting SRE style DevOps practices in smaller organizations. Let me explain.

Slide 18

Put yourself in the shoes of Beatrice, a DevOps engineering working on a small application that profiles information on all the artisans in Portland (and we established - there are a lot!). Starting out you're just trying to find product-market fit, you aren't interested in building a platform. You don't have a robust observability system, you probably have a handful of runbooks you share between your small team. You probably started using some noOps platforms but as you've scaled you've needed something flexible to support your more complex needs. The gap to create a platform and focus on automation is far too high to delay critical feature needs.

It turns out even if you used immutable infrastructure as code principles, automation doesn't appear from nothing fully mature. For beatrice, even in a greenfield application platforms are unlikely to appear at all. Now, brownfield applications transforming to DevOps are even MORE difficult. Creating a platform is no easy undertaking and takes significant new investment to make practical. Outside of Silicon Valley unicorns, the availability of that investment is rare. Clearly there is a need for a guide to help us wade through these murky waters.

Side note - Van Gogh's painting pictured here is called "Green Wheat Fields with Cypress." That's why I chose it - it's regarded as one of Van Gogh's more balanced, calm paintings that "doesn't manifest psychological tension" as an art historian described. It was also painted while he was at an insane asylum. So maybe greenfields aren't as calm as we'd thought.

Slide 19

So we've talked about the conflict. Let's keep the argument going. I'm sure there are some music hipsters in the room and we all know there is a time honored tradition of one-ups-man-ship amongst music hipsters. What band did YOU discover before anyone else? What band were YOU listening to while they were still playing in their parent's garage?

Mine was the Cure. The sad, sad Cure. I'm just kidding. The album cover pictured here is The Cure's Disintegration for those who aren't familiar. I've been listening to it a lot, on vinel of course, and since I'm trying to transition from the conflict to the cure - it was an easy pivot. Sue me.

Slide 20

So what do we need in a Cure? Obviously balance, between the satisfaction of mastery and a process towards automation. We need something that is both lovable, or fulfilling intrinsically

and efficient and valuable extrinsicly. It has to be a process that can tackel both the rapidity of greenfield and the constrained resources of brownfield. It needs to work for more than just cloud-scale services with unlimited resources, it needs to be actionable for the DevOps team of two.

So here's my proposal.

Slide 21

You've heard of inbox zero. The popular Getting Stuff Done process for managing your inbox. I'm a big fan. I'm not saying we treat our runbooks like our inbox - but I am going to rely on some of the essential components of the philosophy. Those components are completeness - it inbox zero spans all things in your inbox - prioritization - you are required to make quick prioritization decisions to place email in the appropriate bucket - and rigidity - there is a formal process you can follow of small iterative steps to form the basis of a habit.

My proposal for Runbook Zero relies on those same fundamentals - completeness, prioritization and rigidity.

Slide 22

Principle one - completeness. We already version controlled next to everything. I We've rode the way of infrastructure as code, in some places we define our entire CI/CD process in code. You are probably here this week to hear about how to transform your observability system into code, and version it alongside the rest of your applications capabilities. t's common for runbooks to live in source code. These days they are just as likely to be created BEFORE a service enters production as AFTER. Runbooks can take the form of markdown files or executable jupyter notebook styles runbooks. Operational scripts are checked in. It is these best practices which form the building blocks I propose we stand on - but there is one thing missing.

What we need is to instrument the usage of our tools, so that we may honor and improve them. How many times did that script run, how many times did we use a runbook, how often was the runbook successful in resolving the original incident.

We need this kind of data in order to do that one critical step. Prioritization.

Slide 23

Of course I'm a product manager by trade, so it's self-serving to say that the hardest questions are ones of prioritization. I know first hand that WITHOUT effective means to prioritize, and COMMUNICATE importance, projects stall. How frequently does important work get shelved because lack of information means low prioritization. In a response system where there is such information as usage, success-rate and risk - we can easily make prioritization decisions even in-line with prioritizing features.

Bucket your current response mechanisms into level of automation and you'll now have the data to prioritize response improvements alongside security fixes, bugs, features, performance, everything. But if I told you that there is a manual runbook process that is high-risk, takes 2 hours of a team members time, and has been run 10 times in the last month - how would you prioritize it. Absent this feedback loop it would never get prioritized within a team without dedicated operations members.

There is a fliip side example to this. Often times we pursue automation for automation's sake. We might look at a simple script we've written and think - oh that would be easy to automate only to find out that that automation runs once every year.

Slide 24

The final step is that process. Starting from scratch is difficult - but starting somewhere, and taking small incremental steps is critical. The wonderful folks at VictorOps coined a term - minimally viable runbook - I love this term. It also speaks to a spirit of incremental iteration. Even after you've created a minimally viable runbook, think about how you can minimally viable changes to that. Let's say that high-risk process could take 30 hours to automate fully or 2 to significantly reduce the risk. Incremental steps free us from the paralysis of deprioritization.

Slide 25

Let's go back to Beatrice. Her team has a greenfield application serving up information about local Portland artisans. The team has instrumented their response systems and they've categorized where they have manual tasks, scripts and executable runbooks and zero-touch automation. The team can prioritize in an iteration cycle improvements to high use, high impact, high risk activities to add automation, iteratively and incrementally building a platform that is prepared for their scale.

Slide 26

The principles of Runbook Zero are outlined here, instrument your response systems and prioritize incremental improvements. If you are me you are saying this all sounds too familiar.

Slide 27

It does because it is. What we are talking about is the lack of formality, process and systems for the continuous feedback portion of the devops loop. Each new role in DevOps, whether it is testing, security, designers or you can also think of it as each additive non-feature capability of an application - quality, security, design, performance, manageability - REQUIRES a new approach to continuous feedback. How do we know where to improve next? How do we prevent ourselves from being blocked by the dogmatic need to do EVERYTHING. That's what Runbook Zero gives you.

Slide 28

So artisanship and automation appear at odds, and we need to focus on enabling satisfying work for our DevOps teams. We can do that by balancing an approach that enables mastery as a process - the incremental improvement of our response capabilities make for a more available, more manageable application over time and only investing in automation when it is critical.

As observability professionals you've had the luxury of thinking about building scalable applications full-time but in order to really bring SRE style DevOps to all application development teams, we need to find a balance between automation and artisanship - and Runbook Zero is the way to get there.

Slide 29

Thanks for you time! Keep Portland Quirky!

Example - Rob Ot

- Rob Ot has an application called Artisans - he's just started creating it
- Rob deploys the application to product and - of course - something goes wrong.
- He fires up the one runbook he's written and does his best to fix it
- He increments the number of times he's used the runbook, adds his fix and moves on
- While improving his application he realizes that manually re-instantiating the database for his application is risky, and he's already done it ten times
- During his next iteration he moves the written runbook into an automated script
- He then realizes he's been restarting X services 30 different times in response to

Outline

1. About Me - 3 min
    a. Politics
    b. Open-source
    c. OpenStack - Private Cloud
    d. GitLab -
    e. Cara?
2. Artisanship - 5 min
    a. Portland connection
    b. Pictures of portland artisans
    c. Silliness of artisans
    d. Definition of artisans
    e. What makes artisans tick, science behind the satisfaction (book drive?)
    f. What's the opposite of artisanship
        i. Assembly line
        ii. Robotics
        iii. automation!
3. Scripts - 5 min
    a. Everyone hates them
        i. Mechanics
        ii. Doctors
    b. We are not button pushers!
        i. Science behind why we hate this, why this is deeply unsatisfiying - commodification of work
    c. In operations we call these runbooks
        i. We hate them even more
            1. Not only do the deprive us of from being human
            2. They aren't even automated! (wait what?)
4. The conflict - 5 min
    a. Strict runbooks rob us of our ability to be artisans

    b. AND they should be automated away, why have runbooks in teh first place, just start with automation! I'll find my artistry in creating the atutomation.
    c. We heard this time and time again, operations involves people, and because it does so we have to think about people first - then systems
5. The process - 5 min
    a. observability systems don't appear from nowwhere, operations manuals aren't automatically born fully matured
6. The cure - 7 min
    a. Start with a metric, an alert and a runbook
        i. Maybe the alert is "website down"
        ii. The runbook is an initial set of diagnostics
    b. Version control all of this
    c. Start the process - operate it, respond to that alert, respond to incidents not based on that alert
    d. Keep metrics
        i. Error budgets
        ii. Response time - MTTR
        iii. Runbook success rate
        iv. Runbook uses
        v. What is the metric that tells you if the task is good for robots or not?
    e. Add new alerts, helpful executeable tools in the runbook, new metrics as you go
    f. Runbook Zero - work to automate your runbook, it's an example of real world scenarios that you've probably encountered more frequently then others, that is your prioritized list.
    g. Employ your artisanship as part of a system. Use your human artistry for the important things, leave the robots to do the un-important ones.
7. The gain / wrap up - 2 min
    a. Focus on what is most valuable for a human to focus on
    b. Don't suffer from alert fatigue
    c. Iterate!

DEMO -

- Evolution of an App with Runbook and Automation
- Simple ruby application - in GitLab with metric and alert and runbook setup
    - GitLab template
    - Artisans - you can add them or you can
    - Instrumented with Yabeda - https://evilmartians.com/chronicles/meet-yabeda-modular-framework-for-instrumenting-ruby-applications
    -
- Go through an incident, use the runbook, add something new to it (source controlled?)

Abstract

For years, ops teams and runbooks have had a strained relationship. The teams understood that runbooks were essential but often felt restricted by them. An emerging trend is causing a shift in that relationship. The future of runbooks is not a space where automation and artisanship are at odds, but where the artisanship lives within and improves the automation.
In this talk, I'll share our experience at GitLab leveraging Jupyter Notebooks and Nurtch to create runbooks with executable buttons to click-to-deploy the recorded best practice. When complex problems occur, SREs jump in to solve the issue and then update their applications and infrastructure to prevent similar incidents in the future. In effect, SREs craft solutions before problems start; building predictive, self-healing applications.
Your ops teams and SREs can do the same for your applications! I'll show you how with some real-world demos that draw from my experience working on OpenStack infrastructure at Rackspace and leading the Ops product management organization at GitLab.