**# [SLIDE] The Wall Dev2DevOps**

* Hello and thanks for coming to Dev to DevOps.
* My name is Dinshaw.
* I work at Constant Contact as a developer and I have very little to do with operations.
* The closest I have come to deployments is complaining when they don't go smoothly.
* Until very recently, the job description of a developer rarely said that we need to know my way around our production environment.
* This is a problem. A common one
* But our industry is starting to realize that the separation between the roles of Development and Operations, the 'wall' that we as developers may or may not 'throw our code over', is not a great way to do business and that it is time for a change.

**# [SLIDE] Law of the Instrument**

* Law of the Instrument Abraham Kaplan, American Philosopher 1918-1993 in 1964
* The manner in which we approach our problems is entirely dependent on the tools at our disposal.
* No tool or the wrong tool? Jobs can be intimidating.
* Perfect tool; jobs look fun.
* Got a Hammer?, everything looks like a nail.

**# [SLIDE] Nail**

* Nails aren't that bad.
* I know how to use them
* They work
* I can deal with this problem.

**# [SLIDE] G'Tool (Guitar-tool)**

* Does anyone recognize this?
* It’s the only tool you need to change the strings on a guitar.
* So then if all I have is a G'Tool, do all my problems look like a guitar?

**# [SLIDE] Flaming guitar**

* Guitars are fun!
* I like this problem...

**# [SLIDE] Angry Monster**

* But what if our problem looks like a flesh eating monster?
* Our problem is scary!

**# [SLIDE] Half-Monster, Half-Computer**

* Now?

**# [SLIDE] Monster Computer**

* Now?
* That’s right. This IS your problem; its all of our problem.

**# [Slide] - Hammers**

* We are going to need more than one tool…

**# [SLIDE] Tree and saw**

* We need the right tool for a big job

**# [SLIDE] Tool tray**

* We need the right tool for a complicated job

**# [SLIDE] Bat Belt**

* We need a tool that makes this job cool and fun

**# [SLIDE] Computer**

* Enter DevOps
* DevOps is referring to the mindset and the flexibility that finds the best tool for the job of delivering valuable software.
* So that we can accomplish all of the extremely complicated, interwoven tasks that are our daily work, and have fun doing it.
* And because it seems that the same tools that create our problems can be used to solve them, I am reminded of the immortal words of Homer Simpson…

**# [SLIDE]- Cause of and solution to all of life's problems**

* Substitute ‘Computer’ for alcohol, and you have our situation pretty well summed-up!

**# [] A Brief History of DevOps**

* At the dawn of the computer age, there was no distinction between dev and ops.

**# [] Colosus**

* You mounted tape
* flipped switches
* turned knobs
* and apparently wore a suite…

**# [] Punch Cards**

* And then came the punch cards; that you handed over to an 'operator' who then gave you your results
* Probably the next day
* On paper
* In a cubbyhole with your name on it.

**# [] Computer**

* And then finally, the arrival of the PC completely separated operation from user and the modern 'IT Operations' culture was born, and the wall began to go up...

**# [] The wall goes up**

**# [] Patric Dubois**

* The late 2000s
* Patric Dubois, Ghent,
* Couldn’t figure out the real one so this is his twitter avatar
* Belgium wanted to work in every possible area of IT

**# [] Velocity Conf**

**# [] 10+ Deploys Per Day: Dev and Ops Cooperation at Flickr**

* Two guys working at flicker
* John Allspaw, Operations
* Paul Hammond: Development
* The community knew there were problems, but these guys really called it out and gave it a face, if not a name, and showed a path to salvation.
* They illustrated the problem with this now famous slide

**# [] StarTrek**

* That there was, and still is, a rift between developers and operations. There was a lack of trust and a lack of respect. And it was self-perpetuating.
* Developers job is to add new features; Operations job is to keep the site stable and fast.
* The two seem to have an innate contradiction

**# [] Devs who think like Ops / Ops who think like Devs**

* This was the beginning of the DevOps movement

**# [] Dubois missed it!**

* Complained about it on Twitter
* People said you should start your own Velocity conf
* Dubois organizes DevOpsDays 2009 in Belgium - Ghent

**# [] DevOpsDays Logo**

* Awesome hand drawn logo

**# [] DevOpsDays**

* Talk titles
* All fairly new ideas then, but still very relevant today

**# [] #devops**

* Twitter dropped the 'Days'
* A lot of people started contributing their experience and ideas
* Grassroots movement sprang up against legacy tools and culture
* From practitioners, for practitioners

**# [] No such thing as a DevOps team**

* It is important to point out what DevOps is not
* Not a spec, standard, or job-title

**# [] #devops**

* People collaborating on common problems, finding the best solution
* Twitter still has lots of valuable

**# [] So what's the problem?**

* What is it about my life as a developer that is so bad that I need the DevOps revolution?
* As a professionals in the software industry, our job is to:
  + Deliver value to our end users by innovation
  + We are supposed to be conceptualizing and building the future of the way people organize their lives.
* But the reality is that we spend a great deal of time on:
  + Unplanned work
  + Configuring and fixing our tools and environments
  + And trouble shooting the hand-off to from Dev to Ops
* The better we get, the more complicated our tool chains and environments become, and fragility creeps into projects and lives.
* Critical systems cannot be fragile
* **Critical systems cannot be fragile**
* Really well put, right? I didn’t write it. I wish I did. Which brings me to the other seminal talk on DevOps…

**# [] Why We Need DevOps Now**

* Benchmarked over 1,500 organizations for 14 years
* All companies, not just SAAS products, have a huge dependency on IT, and the ones he studied seemed to have a leg-up on the others
* This work led him, inadvertently, to the heart of the DevOps movement

I'm taking a lot of things from this presentation. There is a link to his talk at the end of my deck. You should watch it

**# [] High Performance Organizations**

* Etsy, Netflix, Facebook, Amazon, Twitter and Google
* Using Version control and automation to ship [...] applications quickly without disrupting service
* [] deploy code 30 times more frequently than their peers
* [] completing those deployments 8,000 times faster (1 min deploy vs 1 week deploy)
* [] 50 percent fewer failures.
* [] restoring service 12 times faster
* Those are some pretty compelling numbers

**# [] The Dream**

* The dream is that, all day, all we have to do is 'our jobs'
* That our work moves from left to right, through our entire system, business–requirement to production, with very little exception.
* DevOps refers to the ways and means to keep our systems efficient
* And to enable us to keep applications available while constantly introducing significant amounts of change into production.
* New responsibility now falls on Developers in that we need to own both the development of our software, and also the operation of it.
* Responsibility falls on Operations to provide us with the tools that we need to do this.
* Responsibility falls on everyone to start a dialog. To begin to collaborate and to cooperate.

**# [] Holding hands**

* Because, where we used to say at the end of every sprint we have working code AND the env it runs in
* And this collaboration and cooperation is the 'Culture' of DevOps that, as you may have heard, is going to save us all.

**# [] Culture**

* What's that you say? But culture is not going to help you with your current problem.

**# [] Angry Monster**

* You are totally right

**# [] Forget Culture**

**# [] Give me tools**

* Tools I can use, please...

**# [] G'Tool**

* whoops...

**# [] Computer - Common tools**

* 'Common', because if we are all using the same tools we will get better at them quicker,
* Because a common toolset fosters the collaboration and cooperation that will lead to the trust and respect we need to start tearing down the wall

**# [] TEAR DOWN THE WALL**

* I'm will talk about some specific tools in a moment, but one example of common tooling is that developers and operations, and really the whole organization, should be using the same version control system.
* If something happens in the middle of the night, and everyone knows where to look, has access, and knows how to use the tools to say: do a version bump, then one less person gets woken up, one less person is grumpy the next morning, and everyone wins.
* In the same way we refactor legacy code, we now need to refactor our legacy organizations and the way we work.
* When we refactor code, we succeed when it is easy to change the code without breaking anything.
* This is important because our job, as a developer, if you had to sum it up, is introducing change.
* We need systems that are capable and comfortable existing in a constant state of change.
* This is so far from where a lot of organizations are today, that in the Why We Need DevOps Now talk, Gene Kim says refers to this as …

**# [] The Danger Zore**

* You need a culture that keeps pushing into the danger zone
* And has the habits that enable you to survive in the danger zone.

**# [] Scott Cook**

* Intuit's Scott Cook at the Economist conference 2011 Talking about when TurboTax:
* By installing a rampant innovation culture, they now do 165 experiments in the three months of the season. Our business result? Conversion rate of the website is up 50 percent. Employee result? Everyone loves it, because now their ideas can make it to market.
* Compare to traditional retail site

**# [] What is this shift going to look like to me as a developer?**

* At the bottom of our daily stack, is our workstation
* We all have our development machines setup differently. The software we use and our configurations are the tools of our trade.
* And in the same way that a chef has their knives, this is a very personal thing. We put a lot of time into getting everything just the way we want it.
* This is important because it directly effects our state of mind, and our emotional state, when we sit down to work.
* It is great that we have this flexibility, but it can also cause problems
* We've all been ready to push code on Friday afternoon, we merge in master, installed our dependencies, we start our tests, and something blows up

**# [] Fail.**

* An hour later, if you're lucky, you realize it is because someone added a gem that doesn't build with our out of date GCC
* This is a trite example but the problem is clear:
* Discrepancies between one or more developer environments have needlessly cost us time and energy.
* Does anyone here have a script that they use to set up their dev machine?
* A lot of us have probably toyed with the idea of a setup script to provision our personal computers. maybe a shell script, or Chef, or Boxen, an introduction to which is coming in a couple of slides, and even if we haven't done it, it should seem like a reasonable idea because we will probably want the same tooling on all my machines. We can tweak it when we need to
* Now what if someone suggested that you share a setup script with your team? Or possibly your entire organization?
* Sounds crazy, right?
* Unless…
* Unless this script was well written, over-ridable, extendable, and in available in version control.

**# [] Puppet Manifest**

* Then, we can login, read it, fork it, add our customizations on top of it, and fire it off knowing that I am going to get the benefit of consistency with all my colleagues
* That version bumps and other common stuff will be pushed out by your organization, but all your personal stuff will be just the way you like it.
* This is where we are trying to get to. All the benefits of the collective wisdom, with zero sacrifice of individuality.

**# Project Environment**

* OK, so in the perfect world i just described i pull down my code knowing that my system-level tools are the same as everyone on my teams, install my dependencies, run my tests, and everything is still broken...

**# [] Complete fail.**

* Turns out someone added Redis to this project. This is not a system level tool, it's a per-project dependency.
* Its also not a hard one to debug these days, but that is not always the case.
* The point is that Fetching your code and installing your gems is not enough for any real–world project

**# [] Boxen project recipe**

* Just like Server definitions, there can be project definitions that install the proper runtimes and data stores and fetch the latest code
* do a better job of ensuring consistency across multi–developer teams

# CI and Production

* I am guessing everyone in this room has experienced problems promoting code from a development environment to CI and Production, so i will skip the example here.
* The take away here is that these are issues that we all see and accept as 'part of our job'
* The reality is that they are avoidable to a large extent.
* If your development environment and your local test environment are identical to CI and Production, then the time we spend troubleshooting these issues is minimized,
* maybe more importantly, the frustration and the finger–pointing 'hey it worked on my machine, it must be your fault' is also minimized.

In addition to professional and personal satisfaction that we will get from starting down the road of DevOps we also keep ourselves viable

**# [] Working is the new rich**

* We are all incredibly fortunate to find ourselves in a seemingly recession proof niche of our industry.
* DevOps is where our industry is heading. Our job description is changing and it is up to us to answer the call.

**# [] State of DevOps Report**

* LinkedIn keyword 'DevOps' up 50% from 2012 to 2013

**# [] What is a DevOps engineer?**

* slides

**# [] What are we waiting for?**

* Why aren't we doing all this already? Been around since 2007.
* Why isn't everyone doing TDD and CI? Everyone agrees its the right thing to do.
* Well, the easy answer is:
* No time! Too many other priorities...
* But there is another answer that is not as easy to say
* We are paid to be professionals, and in some cases, experts. To know what we are doing. And for the most part, we do.
* We also have a great deal of learning baked into our day-to-day.
* We learn new libraries to use in our code-bases all the time.
* But learning a new set of tools and a new way of working is going to take a little more effort.
* And even for the best of us, it can be a little overwhelming think about being a beginner again.

**# [] Overwhelmed**

* Muddle story…
* This is the right way to feel. It means that you are pushing the edges of your comfort zone and that you are learning. Dont be afraid to hit enter...

**# [] First steps**

* So ‘I’m ready‘ you say. Where do I start?
* One, of the two similarities of the High performing organizations that Gene Kim cites is Automation; the other one being version control.
* Automation is a great place for any developer to start
* Pick one small, familiar thing that you already know how to do, and automate it
* Make sure your editor removes white space when you save
* This might be a simple as setting in your preferences, but that is automation and if it works consistently, then it is valuable forward progress.
* Set up .dotfiles
* Setup a task to rotate your development logs before your machine runs out of space
* These things might seem trivial, but when we start to automate away the little things that take our time, we start to expose the next level of opportunity for optimization of our workflow.
* You will start to see opportunities to automate everything
* Automate the setup of your development environment, the way you like it. Don't worry about if it can be shared at first.
* Automate the setup of your production environment; and then do it as your development workstation.

**# [] Puppet & Chef**

* Slides then…
* Run it often
* Security patches pushed out via a pull request – Boxen Safari – java issue

**# [] Puppet**

**# [] Boxen**

* Github wanted to make sure that everyone could push code their first day
* Works out of the box, but still some kinks

**# [] Chef**

* [Mitchell Hashimoto](http://engineers4engineers.org/#hashimoto)
* Creator of Vagrant
* 2pm

**# [] TDD**

* So once we have used automation to help ensure that our machines works the way we want them to, we can start thinking about stuff a little further up stream like, 'does our code work the way we want it to?'
* The time has come to be ashamed, if we are not doing TDD
* I cannot even imagine working with untested code anymore. I would pull out my hair and then lose my mind.

**# [SLIDE] If you tell the truth...**

* Every line of untested code that we write, is like a lie that we tell to our boss that we forever have to remember. This is no way to live and its no way to work.
* Dev & Test are no longer separable.
* Not doing TDD? You will NEVER get to CD.
* There is no room, no time, for a manual testing step in an automated deployment workflow.

**# [] What we need from ops**

* Ops must provide a one button environment that you can be 100% confidant is a clone of your test production environment. This becomes possible when setup of these environments is scripted with something like Chef or Puppet, and under version control
* One button deploy: we have to know that deployments are happening the same way every time.
* When every deployment is done differently, every production environment is different and no mastery of procedure or configuration will ever be achieved.

**# [] You build it; you run it.**

* Developers do the deployment
* Developers own uptime for code

**# [] Google SRE**

**# [] More suggestions?**

* Pair-Programming
* Maybe even from someone from the other side of the wall
* Have coffee with an operations person
* You will find that many of us have not made this jump yet, bet we are all excited to start.

**# [] Choas Monkey**

**# [] Twitter Murder**

**# [] Chat Ops**

**# [] Amazon @ O'rily Velocity Conferance - June 16, 2011**

**# [] Links**

**# [] Closing**

* Thank you for you time.
* I will be around all day if anyone feels like chatting.