10 Questions to Ask About Your CD Pipeline

Anton Weiss

Hello everyone. Thanks for coming. It's really great to see so many smart folks interested in.. beer. And tech and music, of course. I think it's great - this marriage of tech and music. I've been building and delivering software for the last 15 years and I've been writing and playing music for the last 20 years - and they do have a lot in common. In fact - jumping forward to the subject of my talk - building a great Continuous Delivery pipeline is a lot like arranging a musical composition with a lot of instruments. So.. what am I going to talk about? This won't be a workshop or a use case. I love learning about new smart hacks just like any other techie, but I think conferences like this one are especially great for sharing our visions on where the industry should be heading, on how we want to work together and on what are the things that annoy us in our everyday professional life. So I decided to follow James Altucher's advice. He talks about coming up with 10 ideas everyday as a means of developing your creativity muscle. I've been building CD pipelines for the last 8 years and I wanted to see how hard it will be to come up with 10 good questions about them. I think some of these questions will resonate with you as well, still others may seem foolish or too obvious - but what I'd really really like - is to get your feedback on this. Because I do have my answers to these questions, but yours will surely differ. And more importantly - I'm sure you'll be able to come up with many more good questions. No I'm a firm believer in continuous improvement - and asking questions is the only way to keep getting better.

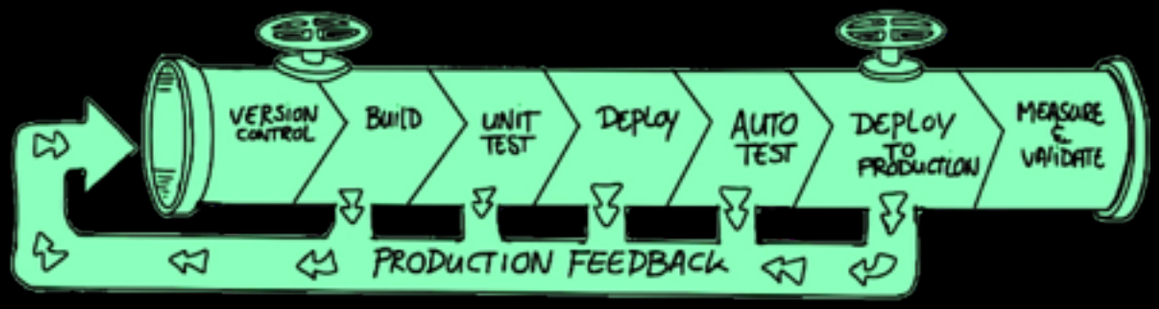
Continuous Delivery

- **Continuous Delivery** (**CD**) is a software engineering approach in which teams keep producing valuable software in short cycles and ensure that the software can be reliably released at any time. It is used in software development to automate and improve the process of software delivery.

(Wikipedia)

This was a long enough introduction so let’s get to the subject. First let’s remind ourselves what CD is. I didn’t coin the term so I’ll just reference the Wikipedia definition. “” So this is all great and sounds fantastic, but how do we achieve this in real life? What is the vehicle that takes us to the wonderful world of CD? The CD pipeline of course: all the processes, the automation and the infrastructure required to make CD possible. I to not be too abstract I brought a picture:

CD Pipeline

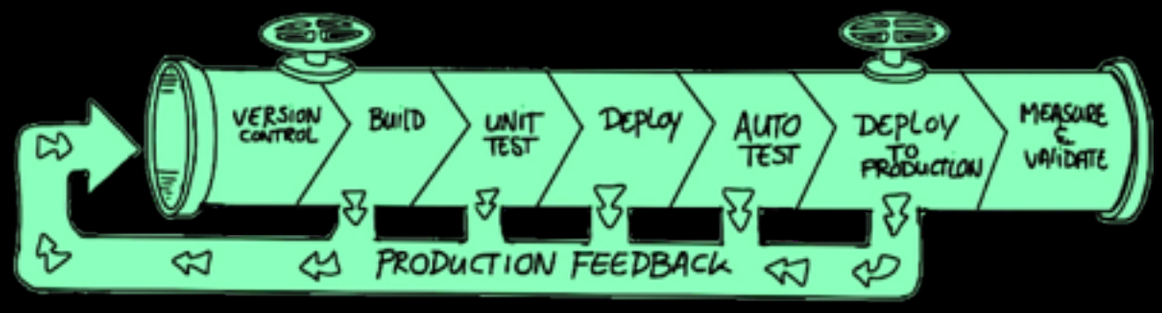


I stole this from xebialabs site. I guess I could've drawn it myself but I've been working in software long enough to realize that in most cases it's best to use things I downloaded from the internet and customize them to fit my needs.

So this is the proverbial CD pipeline - enabling the flow of our changes from the stage of requirements through source control, builds, tests and deployment into productions. Or as John Willis puts it - from "A-ha" to "Ka-ching". Now we have a picture - so let's start asking some questions.

0.

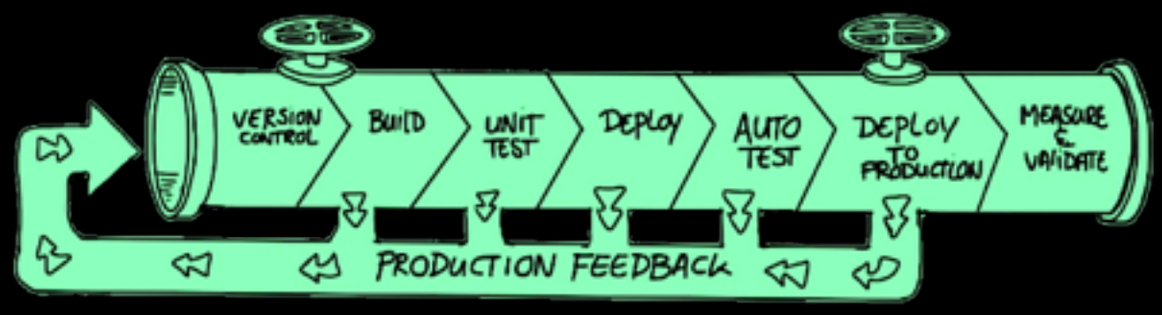
Do we need Continuous Delivery?



BTW - I have to admit - I came up with much more then 10 questions and I had to throw away a lot of waste. Still I wasn't totally sincere when I called this 10 questions. You can see I started at 0 - as any hacker would do, but the list does go to 10 and not 9… It's because this first question is mostly philosophical these days. Do we need CD? It's year 2015 - this is the way things are done today. Or at least this is how almost everyone agrees they should be done. And still - only 5 years ago this was the privilege of the chosen one. So I think it's good to be critical and ask - are we going in the right direction? Is CD really the right way of delivering software? Some may think, for example that it's only a good fit for web scale projects. I can talk from my experience - I'v been enabling these processes for all kind of systems - mobile development, embedded microchip software, enterprise systems, storage solutions.In all of these cases the initial investment paid off in improved quality, faster time-to market and most importantly - in enhanced collaboration.

1.

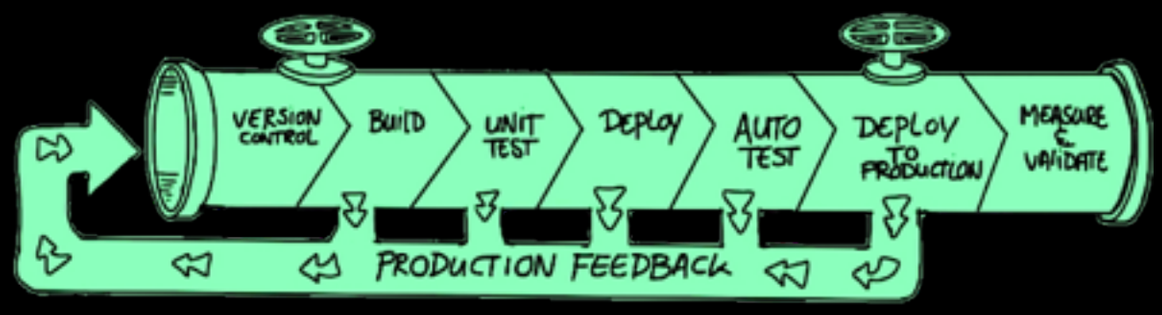
Is it Making Lives Better?



And this takes us to our first real question. Is our CD pipeline making lives better? For me - this is the most important question. Yes - we want to be agile, efficient, we want to have great quality but we must always remember that the most important asset of any software shop are its people. And people need to be creative, people need to have good tools, people need to have access to whatever information and resources they find necessary in order to deliver value to the end user. So the first thing we always need to ask - is the pipeline actually making lives better? Are people really using the great automation we build? You know - I've seen so many sophisticated automation solutions stay unused for months and slowly disappear into oblivion but on the other hand I've seen very simple tricks providing so much value and making developer, qa and ops so much happier. In short - let's always make sure we're building things people really want.

2.

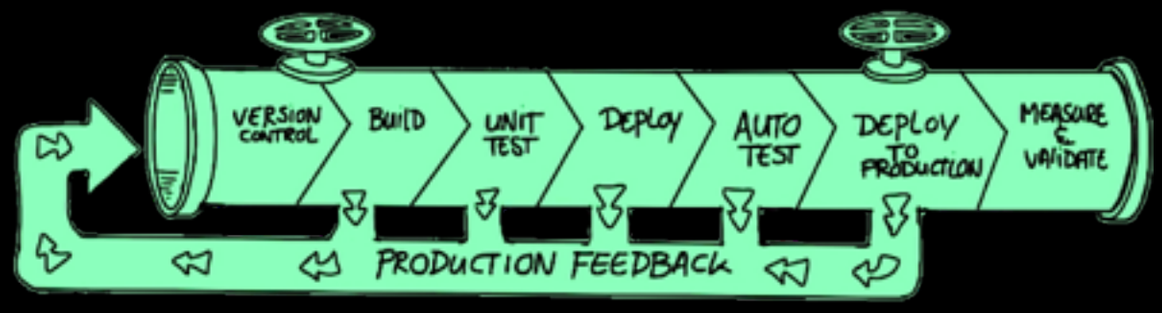
Does it Hurt?



Now with all this happy-happy, peace and love mood the next question may seem a bit counterintuitive. Does it hurt? Yes - I certainly believe Continuous Delivery should hurt. I’ts like they say in sports - no pain, no gain. Jez Humble and Dave Farly - the godfathers of CD have put it this way: if it hurts - bring the pain forward. CD is there to help us improve, so if it hurts- it’s a good thing - let’s just make sure it doesn’t always hurt in the same spot and we’re actually resolving our pain points and improving our flow.

3.

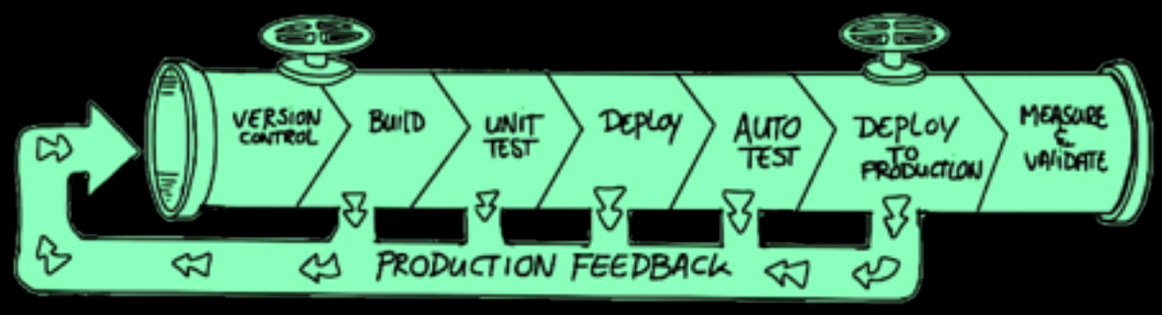
Is it actionable?



And this brings us to the next question. So things fail - we realize we have bugs and integration issues and - and we always do. So the question is - by looking at the pipeline - can we always say what went wrong? Is the output clear enough? Can our users - be it dev, testers, ops, or product understand the problem and find the way ot resolve or work around it? Because if not, if they get stuck each time something fails - this isn't continuous anymore, this isn't really the level of self-service we want to have.

4.

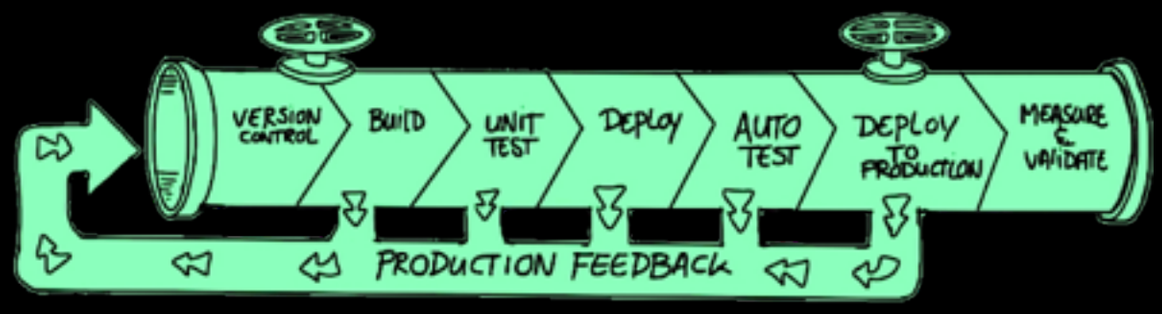
Are We Managing Change (in the Pipeline)?



Next question. CD is about delivering changes. And many times these changes will require the pipeline itself to change. And you know how it is - something unexpected entered the pipeline - everything breaks and we need to fix fast, to adapt to the changes. So now our pipeline is different. And then - in a couple of months we find a bug in that old version, and we need to rebuild it the same way it was built. If we don’t manage the change in pipeline configuration - this wont be easy at all. It doesn’t matter how we do it - taking VM snapshots, using containers, managing our infra as code - the main thin is to always make sure we can go back and recreate what we did back then.

5.

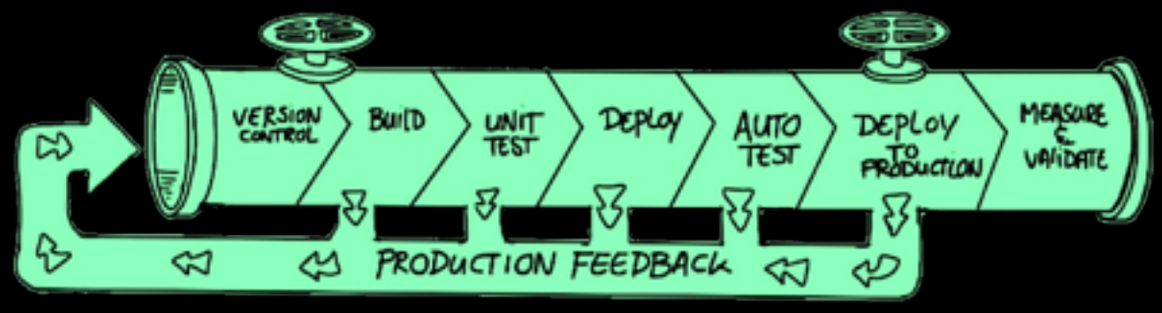
Is it Enabling Communication?



Once you the CD pipeline in place - it becomes the beating heart of your delivery process. Everybody should be using it, and when many folks are using the same technology - they need to have a way of communicating around it. Github engineering provides a good example of how this can be enabled with their use of ChatOps. Having a company chat is a great thing in itself - make sure you try this if you haven't. Use hip chat or slack or any other solution. And then - make sure your CD pipeline is wired into it. Read the Github engineering blog posts for more details.

6.

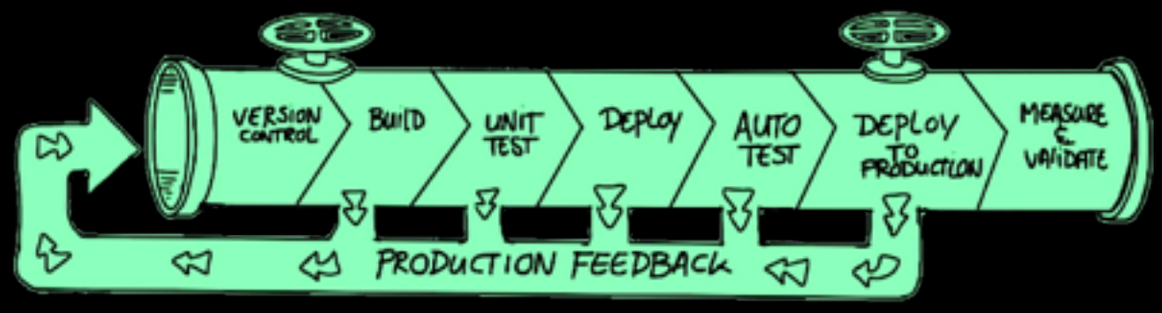
Is it Hermetic?



Another point that sometimes gets overlooked. Is out pipeline hermetic? Are we sure nothings leaks out, and not less importantly - do we have the total control of what goes in? Nowadays we all use 3rd party libraries, docker images, vagrant machines, modules and so on. While developing -we must have all this up to date and available. But when delivering - we must make sure we don’t rely on any outside resources that are out of our control. Make sure we don’t pull from maven central, or directly from github. Continuous Delivery is complex as it is and we don’t need any more elements of chaos potentially ruining it for us.

7.

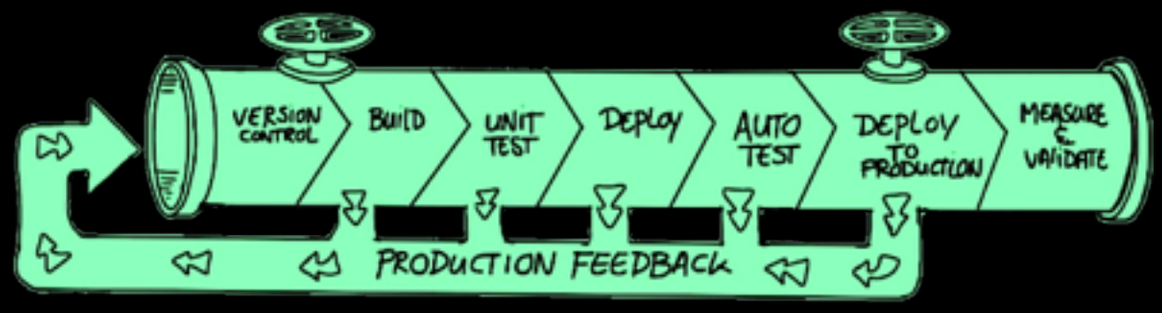
Are We Measuring?



This may seem trivial - but it does get overlooked. So we have a CD pipeline. Can we tell just by looking at it where the bottlenecks are? What should be the next thing we can improve? Listening to our users is vital, but measuring is equally important.

8.

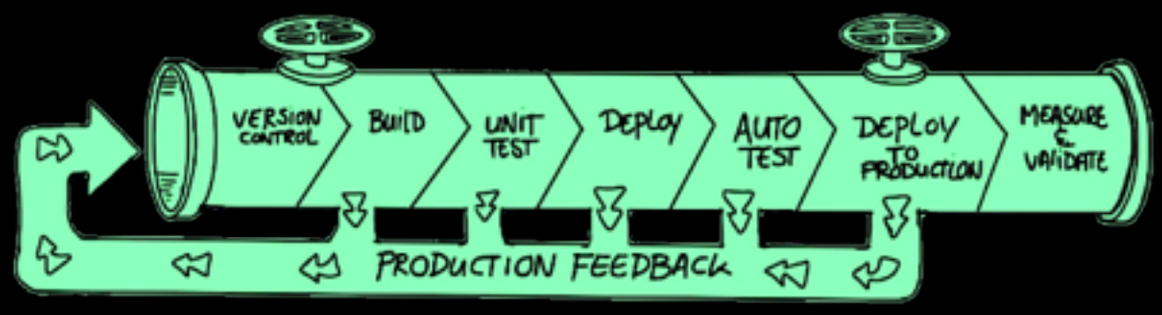
Is it Transparent?



So I was talking about clear output, about failures and reports being actionable. But transparency is more than that - transparency is making sure that everybody is able understand what is going on inside the pipeline. First of all - this is the only way to get some real feedback and then - we must remember - this is built or tech people. And they usually don’t like to just push buttons - they are always curious to understand how it works inside. Let’s satisfy this curiosity by making everything as visible as can be.

9.

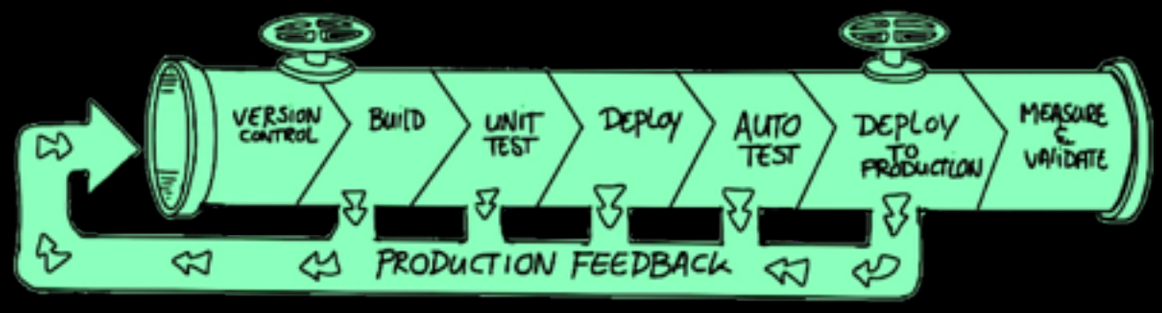
Does it Provide Information or Data?



The pipeline doesn’t only deliver change - it also accumulates a lot of important data in the process. Our commit messages, package versions, environment configurations, release notes, logs. There’s been a lot of posts lately about using big data processing to make sense of all of this data. But even before we delve into these solutions - we can make sure we collect the bits of data we already realise are important and interconnected and convert them into valuable release information. This information should be easily accessible by all in once central place - and we should be continually improving it according to our end users real needs.

10.

Who Owns the Pipeline?



And the final question: who owns the pipeline? In too many cases what we see is tat there's a DevOps team owning the pipeline and all the rest are either enjoying it or complaining about it - depends on personality type and the quality of the pipeline of course. Needless to say - this is terribly wrong. Is everybody is using it - everybody should assume ownership over it. I'm not saying everybody should be able to modify things without any supervision - after all there's some special knowledge involved - and that's how I earn money :) I'm just saying the decisions on how the pipeline should function should be democratic, we must provide teams with as much control as possible. After all as I said - the pipeline is there to make our lives better, and democracy is the best way we know to make sure the majority of us is satisfied with how things work.