Automation of Testing, Configuration & Deployment

by Edmond Meinfelder

Hello! In this presentation, I'll be talking about "Automation of Testing, Configuration and Deployment". Which is something I've come to care about because it makes my job so much easier.

I care about automation of testing, deployment and configuration because it makes my job easier. Mainly, I come in to projects after things have gone wrong. Typically, the first thing I advise is writing & automating tests, configuration and deployment, because it has profound and immediate effect on projects in crisis.

Brief Bio

BSc & MS in Computer Science

Over 20 years experience writing software in online environments

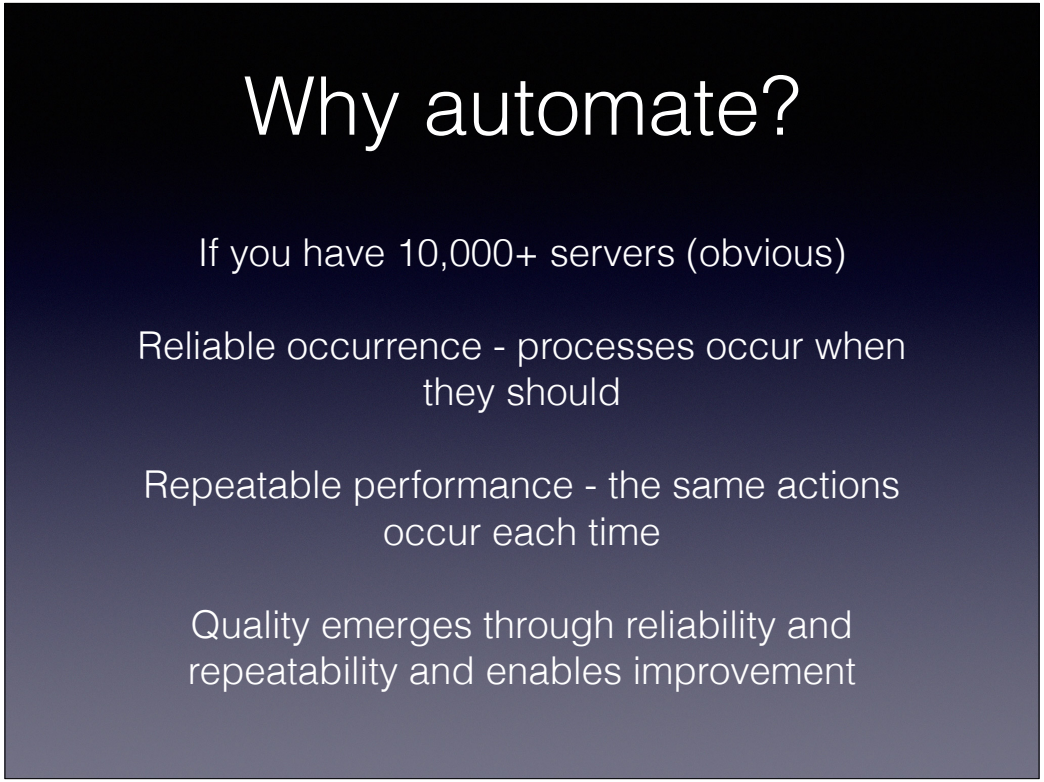
Worked in small series-A startups and publicly held international corporations

Currently working as a consultant at Panasonic

Who am I and why should you listen me? I have studied computer science and went on to get a masters degree because I enjoyed it so much. Really. The passion matters more than the degrees, in my opinion.

I've been writing software, managing teams writing software or consulting on software projects for over 20 years. I've been in the trenches. 20 years in online? Yes, that pre-dates the Internet. Well, the web-based Internet as we know it, but not IP-based protocols.

Right now, I'm consulting at Panasonic. They were nice enough to let me take the time to be here today and are proactively taking steps to avoid a lot of the problems I'll be talking about. They've been and are wonderful to work with.



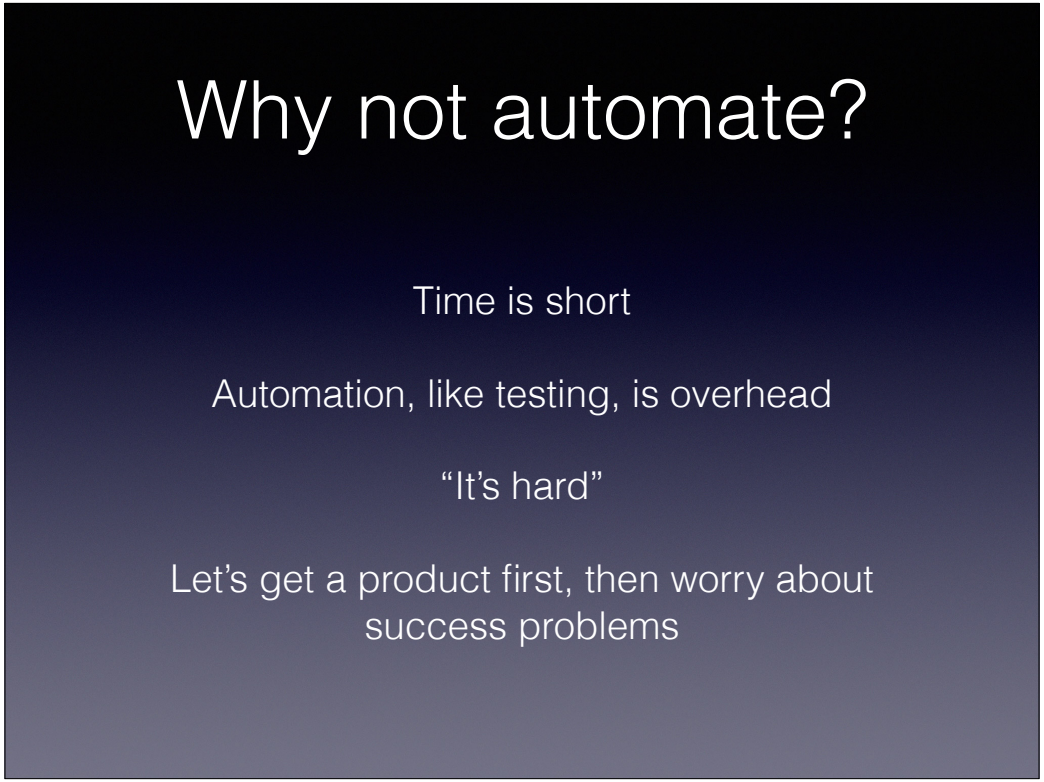
For many, the answer is "to save money " This is obvious if you have 10,000+ servers. And I've seen it done. However, the majority of cloud-based applications require less than a dozen servers. Many require only 1 or two servers and these applications suffer poor quality.

Over the years, I've seen both large and small development environments. Regardless of size, I've seen a surprising lack of automation with testing, deployment and configuration. The theme is: automation exists if the task can't be done manually and if something can be done manually it is.

Reliable occurrence - The majority of projects need a few servers, and now we use cloud providers. With cloud providers, we create an OS image, install the needed software and then just re-use that image. And if you need to apply a security patch, well you just do it once per server. But what if, before, the last server is patched, the administrator is distracted by some emergency. The patch on the last machine is never installed.

Repeatable performance - What if the tests are run on each developer's machine before release to an online service? Imagine if Joe the programmer has an old version of a module globally installed on his system, which passes the test while in production. However, the latest module is downloaded in production and has a performance issue. Manual configuration destroys repeatability, which hurts quality.

Quality through reliability and repeatability - If the process always happens when it should and is perfectly repeatable then you can accurately observe and measure what you are doing and improve! Sounds easy. Everyone agrees it easy, so let's encourage everyone to do it.



Why not automate? Seriously, why not? I see a lot of projects at small and large companies and many have no automation. None. Yes, I have worked at large companies. I’m recently consulted at a company with a $2B market cap, working on a project that had nothing is automated. This is not uncommon. Though, I’m seeing chef and puppet get some traction with configuration, a lot of work that should be automated is NOT.

True, Amazon, Google and Akamai all have huge investments in automation, but they had no choice. What they are doing would not be possible otherwise. But, when it’s possible to do a task manually, it usually is. I believe it’s because software development is a struggle. We focus on the immediate task at hand and run in a straight line towards that goal. Here, I’m asking in the context for those us having applications requiring a handful of servers, which is most of us.

”Time is short” is self-explanatory. And when you compress software development timelines, quality gets discarded. Hopefully, you have good visibility through process to your team’s decisions on trade-offs with quality versus time, but when time is exceptionally short, quality gets dropped - often invisibly. Automation is a part of your quality effort!

Quality is overhead. This is true. It’s more true for commodity products with little variance in quality, which is never the case with software we are actively developing. While I think the phrase “Quality is free” misleads, it’s true in that quality pays for itself over time. Somehow overhead came to mean non-essential, but it actually refers to fixed costs of operations. Electricity is overhead. You should minimize overhead to be only what’s needed, but I see too many software teams thinking “Automation is non-essential,” but given the low quality and lack of repeatable processes I’ve seen, I disagree.

Product first, then quality through automation is understandable, but only to a point. At the start of a project, it should take about 8 hours automate a manual process done 10 times a day that takes 15 minutes. This year there are 251 business days, that’s just over 26 days of effort, a bit over 10% of each work day. Now, if you spent 26 days to automate a 1 minute process occurring 10 times a day, you’ve lost time. This is measurable!

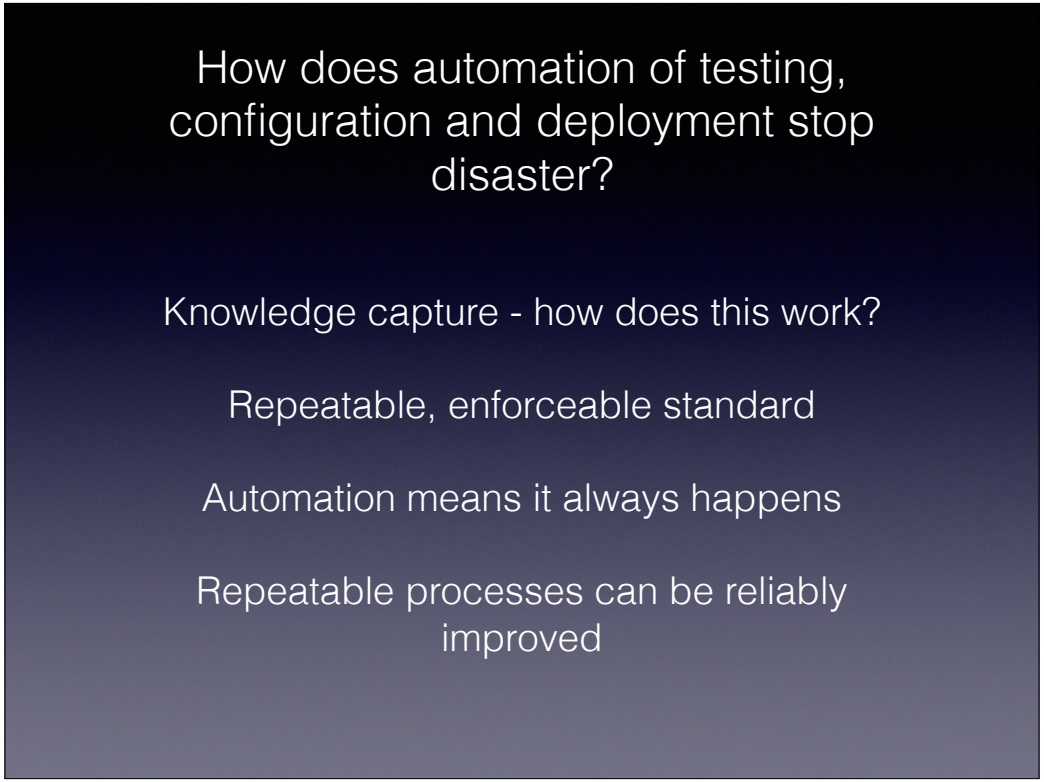
The following really did happen

- Online game costing $15 million to create receives award “coaster of the year” (in the age of CDs)
- Licensed IP game fails over Christmas, 4 managers and 2 teams lose their jobs, including the CEO, VPE and director of engineering, additionally 4 key developers left from “disgust.”
- A company, once competitive in file sharing became irrelevant in 3 years

The following also happened with automation:

- A small team re-invented how a company users downloaded large video files, saving millions in CDN costs
- A small team created a common platform for all others development teams to solve all the common problems each team had to solve again and again, saving millions of effort in the first year alone

These are all from personal experience.

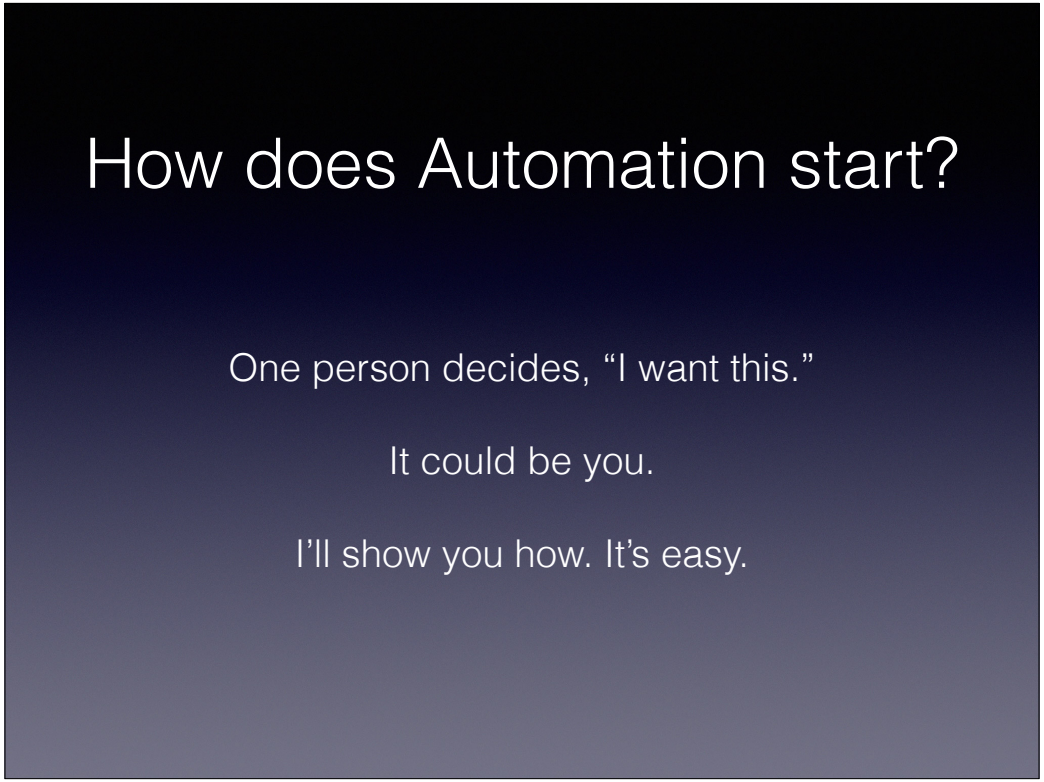


Knowledge capture - To automate a process, it must be captured in electronic form. This makes the steps evident. Maybe not accessible to non-technical members, but that process is now recorded in a form others can see.

Repeatability - The foundation of quality is repeatability. It's through the lack of repeatability that we see errors, but to spot errors, we need to good repeatability. Manual processes are prone to variability. Years ago, Excite@Home was a big deal with email and other services. They had a manual back-up procedure that wasn't written down. One person would tell the next how its done. One day, a guy got it very wrong. Excite@Home's email was offline for days.

Automation makes it happen - Technically, an automated process can be scheduled by a computer and if you design with single-points of failure, then yes, automation does make it happen.

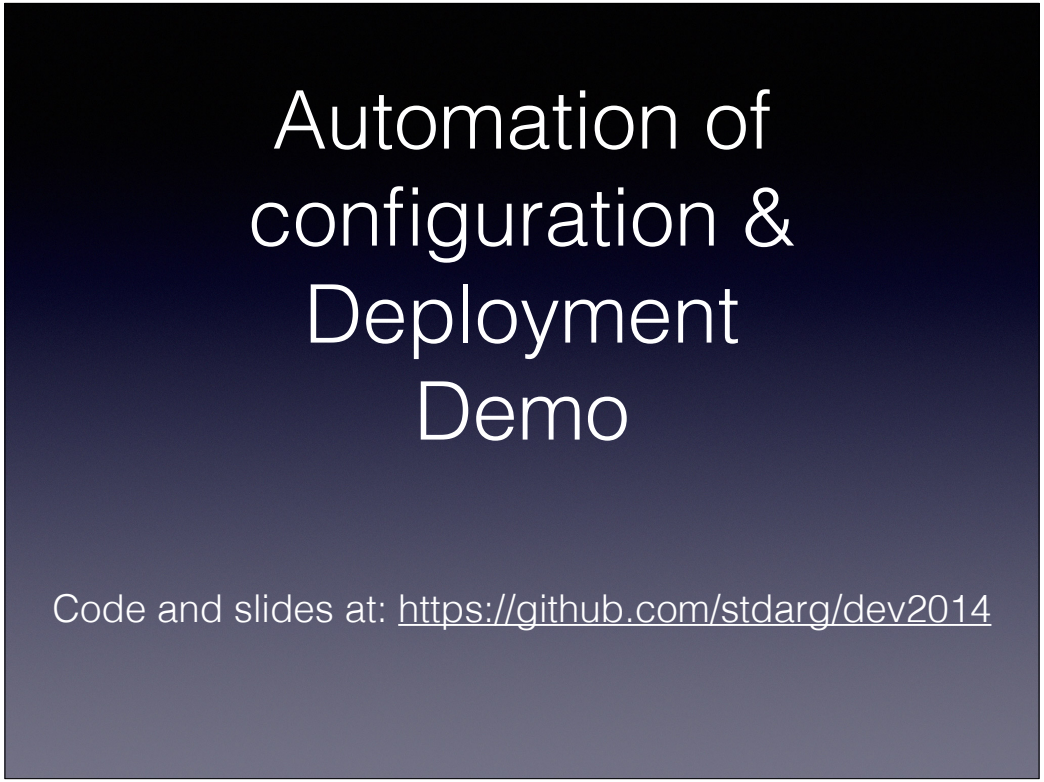
Repeatable processes can be improved - If you are doing what appears to be the same actions but getting different outcomes, you have low quality through lack of control. This is common with manually-run processes done by hand. And when you get varying results with what appears to be the same actions, where is the problem? It's hard to know. However, when you automate each step and run them in a controlled environment, you *almost* always get repeatable processes.



In many companies, it’s **one person decided** it was a good idea and secretly spent a week making an automated test framework. How do I know this? I am usually that person and have spoken to that person a few times. I’ve never seen a management lead initiative to automate at companies outside of these whom you’d expect to need to automation.

Interesting fact, sometimes, **Agile makes it impossible for me to do this**. Why? The great transparency Agile gives on a daily basis can make it hard for developers to pull off such irresponsible stunts. Also, business owners typically are not technical and giving them the keys to row boat manned by a team of programmers is not always a good thing. Business owners are typically product focused, as such non-functional aspects like automation take a back seat.

Regardless, automation of configuration, deployment and testing is easy. Let’s run through a demonstration.



We have a vanilla Ubuntu 14.4 image at dev2014.stdarg.com. When I login it looks well, plain. Well, it's not entirely vanilla, after I installed Ubuntu, I placed my public SSH key in the root account's .ssh/authorized_key file, enabling SSH commands without passwords.

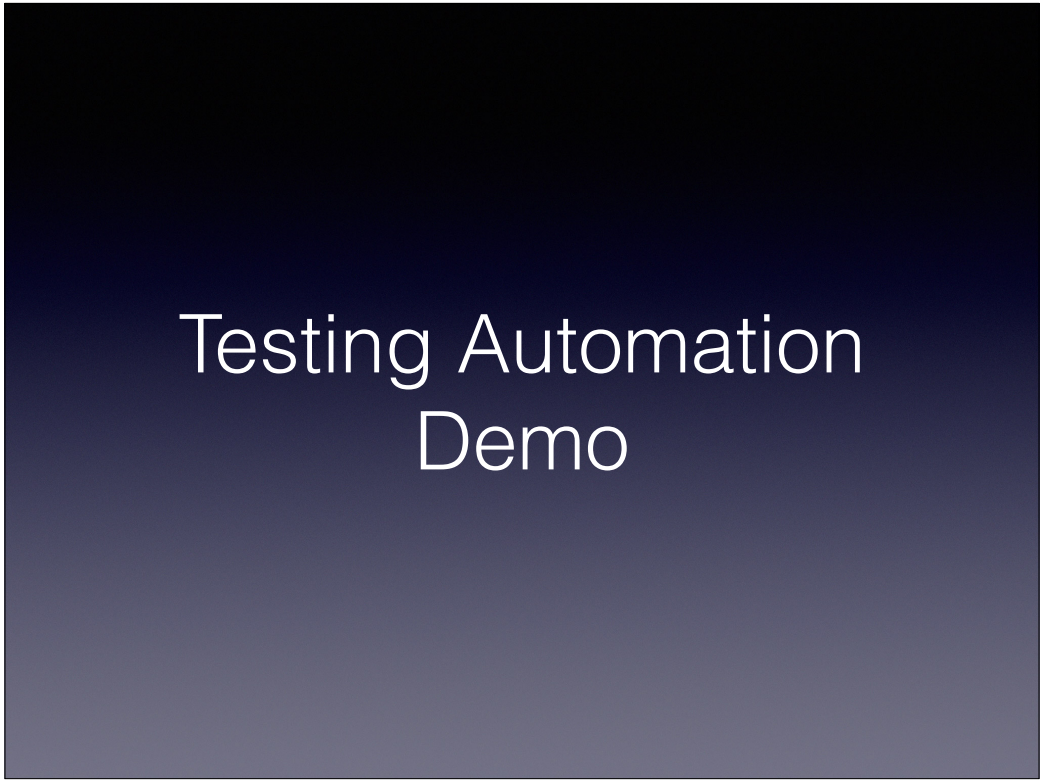
[Show login at dev2014.stdarg.com and how it is not yet configured.]

Still, there's no git, no languages I need or tools. I need a dev environment and maybe have a few files copied over and installed. Easy.
I'm using Ansible, the latest shiny toy from the automation toy box that automates configuration and deployment. With a small YAML file I'll direct Ansible to install: Node.js, npm, git, mocha and Jenkins a popular automation tool for building and testing. Additionally, the YAML file directs Ansible to copy over some configuration files for BASH, git and vim. Here is the command: ***ansible-playbook setup.yaml***

[Watch output from Ansible as it configures dev2014.stdarg.com and answer some questions.]

I installed Jenkins, a popular automation tool for builds and testing. All that from less than 40 lines of a configuration file. Also, I in a few lines, I can update the distribution and install mongodb.

[More examples shown and run if people seem remotely interested.]
[Show new jenkins server running at http://dev2014.stdarg.com:8080/]
[Show nicely configured server from command prompt]
[All of this content is viewed looking at a demo available on github at: https://github.com/stdarg/dev2014]



Here, we checkout a simple project from github and run its test under Jenkins AUTOMATICALLY when a change is checked into GitHUb. We're not automating the project configuration of Jenkins to get a feel for the software. It's easy! However, automating the setup of Jenkins would be good idea.

Manually install the Jenkins GitHub plugin - select install and then reboot.
Create a new project.
Select Git for the repo.
Add the URI path to your GitHub repo.
Add your git credentials :(
Select "Build when a change is pushed to GitHub"
Select email notification.
Add the following build steps:
  npm install
  npm test
Hit "Apply" then "Save"
Add web hook on GitHub repo - and test web hook.

Checkout the is2 project and commit a change with an error.
Commit a change to the repo and the tests will occur automatically!
And the report should show the error.

Thank you!

Any questions now?

For questions later: edmond@stdarg.com