- I’m going to talk about my vision for the future of system administration, and a question for you to contemplate

- I don’t have a crystal ball so I can’t predict the future but everything I’m going to say is based on work I and my group are doing and will be doing for the next 1 – 2 years

- When I started in 1974 I was called a “programmer”
- Not “system programmer”; I was developing applications

- In 1987 I switched to "classic" system administration

- All the usual stuff: backups, O/S upgrades, etc.

- In 2007 I went back to programming but this time I was called a “software engineer” or “software developer”

- In 2012 I started concentrating on software for system administration
- In 2013 I joined MathWorks, in part to work on software for system administration

- Since 1974 there have been **huge** changes in hardware

- My first machine took up half this room, had a 0.27 MHz CPU, 16KB of CORE (memory), and 2 x 0.5MB of disk

- Even the cheapest iPhone is between a thousand and a million times more powerful
- And it fits in your pocket

- There have been similar changes in system administration but they're harder to quantify

- The rate of change for H/W seems to be slowing down
- The rate for system administration is speeding up

- I predict **big** changes in the next 2 or 5 or 10 years

- If your company's product is S/W, I predict the changes will be in < 5 years (and maybe even in just 2 years)

- For everyone else the changes may take up to 10 years

- The software my group is writing will fundamentally change most of “server administration” at MathWorks

- Other companies are writing the same S/W
- Most notable is Netflix

- Eventually there will be “products” to provide the same services for any company

- So what does my software do, and why do you care?

- I feel **strongly** that SysAdmin today lacks real automation

- Puppet et al. are “force multipliers”
- Except for repairing files that change, a SysAdmin still has to cause things to happen

- Think about new servers: A SysAdmin has to create the machine and do enough configuration for it to connect to the network so Puppet can then run

- Networks are worse because there's even less automation
- Without automation this stuff doesn't scale

- Why do I care about scaling creating and deleting servers and networks?

- MathWorks has two large software products and a large handful of web and cloud apps

- Each product and app has lots of tests: unit, functional, system, integration, regression, and performance

- They take a long time to run

- The integration tests are the hardest: they require 5 – 8 hosts, each running a different app, 1 of which has the code being tested

- This is really hard to schedule, and takes a long time, so instead of testing against known versions of the “foreign” apps you test against other unreleased code

- It also means having <n> integration environments sitting around waiting for someone to run a test

- It’s a moving target
- Sometimes bugs slip through

- My current project is to let app developers specify the versions of the “foreign” as build-time dependencies

- Done in Maven (pom.xml)

- My software uses those dependencies to create a private network and however many servers, runs the integration tests, then tears it all down and reports the results

- If the tests succeed, a clean copy of the VM being tested is saved for use by integration tests for the next app

- All of this happens through Maven plugins
- Maven in run by a CI tool that watches for devs to check in code

- The CI tool isn’t Jenkins but it’s similar enough
- We use Maven because it handles the dependencies

- With my software, hosts and networks are **automatically** created, configured, and destroyed, with **no** input from the SysAdmin

- **That's** why I care about automating

- And . . .

- From there I can automate all the tests, **including** tests for changes to our Puppet code . . .

- Same process, including running **application** integration tests, to make sure a Puppet change or an O/S upgrade doesn't break the apps

- The software can also deploy a new server (with an app already loaded) into production; it’s just a few different configuration parameters

- I’m betting that at least some of you are saying “What about <x>?”

- I skipped a **lot** of details

- “Yeah, but what about <x>?”
- Meh

- I write software, and I've been doing it for a long time; I know that I can make my software do just about anything . . .

- So instead of “but what about <x>?,” my attitude is “write just enough code to give the developers *something*, and we’ll handle <x> when it becomes important”

- It’s called “Agile”
- It’s not a big deal but it does require a change in attitude and mindset

- So now that my software does pretty much all the work of managing the servers, what's left for the SysAdmins to do?

- Not much!

- Another bit of software we'll be working on monitors our VM usage and tells the SAs when to order more H/W to run the VMs

- After physically installing the new H/W, the SysAdmin goes to a web page and enters the rack location, power outlets, and network ports, selects the environment, and hits “Go”

- All the configuration of the H/W is handled by my software (using Puppet), because configuring H/W is nearly identical to configuring VMs, just with different inputs

- But what about new O/S versions or patches?

- SysAdmin edits a text file and checks it in; my software tests it and if the tests work, makes the new version available for future tests

- But what about new versions of Apache or Glassfish?
- Not the SysAdmin's problem

- The **developers** handle those!
- **WAT?**

- The devs specify the new version of Glassfish, and that causes all the tests to run

- If the tests pass, the new version is made available for production; if the tests fail, the devs either roll it back or update the app to work with the new version of Glassfish

- Either way, the SysAdmin doesn’t have to care

- The new version can’t be released to production unless the tests pass, and that’s the exact same process as app changes

- The SysAdmin doesn’t care because s/he doesn’t have to care
- I say that’s A Good Thing

- So now, pretty much all the SysAdmin does is rack & stack hardware, edit text files, and once in a rare while, write a new Puppet module

- But is that all?
- **No!**

- There’s a ton of software that does all the real work

- So if I’m the guy that writes that software, am I a SysAdmin?
- I say “yes!”

- So remember I said I had a question for you to contemplate?

- Here it is . . .

- Which SysAdmin do you want to be?

- Do you want to be the one that racks & stacks hardware, enters stuff on web pages, and sometimes edits text files? . . .

- Or do you want to be the SysAdmin that writes all the (cool) software?

- If you want to be the one that racks & stacks and edits text files, do nothing

- Your job will change in 2 or 5 or 10 years

- But if you want to be the one that writes the software, **stop** being a “classic” SysAdmin and learn “software engineering”

- That is **not** just “learn Perl and a little bit about coding”
- It’s not even learning Python or Ruby ☺

- “Real” software engineering means learning algorithms, proper coding practices (for readability and maintainability), proper use of version control and branching, . . .

- Agile (in whatever form), how to use & design & build APIs, design reviews, code reviews, how to write software tests, how to write testable software, how to use databases, . . .

- how to make testing an integral part of the development process (like TDD or BDD), writing software that checks all error codes, writing software that returns sensible . . .

- error codes, and oh by the way, learning languages like C/C++ and Java, plus learning how to learn new languages (Go, Haskell, Erlang)

- A BS in CS isn't a requirement but it's at least worth considering, and it may help you change jobs from SysAdmin to developer

- Then, focus on writing “software for system administration” and eventually you’ll be “the new kind of SysAdmin”

- So, which kind of SysAdmin do you want to be?
- Thank you