Navigating Vendor Nonsense

By Ethan Banks

Nathan sent in the following question:

In this particular field, I feel like younger, less experienced engineers are at the mercy of trying to discern the best path forward by wading through the gobbledygook of vendor BS. Are there any practical tips you can give on how you guys evaluate or educate yourselves on your topics?

gob·ble·dy·gook

/ˈgäbəldēˌgo͝ok/

noun
noun: **gobbledegook** noun: **gobbledygook**

1. language that is meaningless or is made unintelligible by excessive use of abstruse technical terms; nonsense.
   synonyms: gibberish, claptrap, nonsense, rubbish, balderdash, blather, garbage, mumbo jumbo, drivel, tripe, hogwash, baloney, bilge, bull, bunk, guff, eyewash, piffle, twaddle, poppycock, phooey, hooey

I've become cynical over the years with both vendor products and sales pitches. Vendors tend to overpromise and underdeliver, especially with the first releases of new products. Salespeople like to take as much budget as they can possibly get, padding bill of materials with stuff you don't necessarily need.

All too often, I've gotten a build sheet back to review, only to see a bunch of line items I didn't ask for, and couldn't even conceive of where they came from based on the conversation I'd had with the vendor.

I wish this were not reality, but it is. On the plus side, there are vendors who do their best to be honest, and salespeople that won't try to screw you. Some genuinely try to come alongside of your company as a partner and match your business problems with technical solutions.

But not usually. How do you defend yourself against these situations, especially if you're new to networking, struggling to wade through the aptly termed gobbledygook?

The short answer? Arm yourself with knowledge.

1. Know your business requirements clearly.

Don't allow a vendor to sell you a solution for a problem you don't have.

Vendor salespeople are often directed by their motherships to push specific solutions, whether you want them or not, whether they fit the problem or not. Be careful if you feel like a solution is being proposed that addresses issues you haven't communicated.

2. Assume nothing about how a technical solution might work or might meet your needs.

Things you just presume would work often don't. Engineering minds tend to work by assuming that something must work in a particular way or have a certain capability, because for them to not be that way is unthinkable. Never take for granted that a solution will do something. If a certain capability is important, make sure it's there and works as you require.

3. Ask detailed, specific questions of the vendor about the solution.

Salespeople will often gloss over potential issues with hand wavy magic to move the process quickly towards a purchase order. Don't tolerate that. Make them prove answers.

If the salesperson doesn't know the answer, set up a call with a sales engineer or technical marketing engineer. SE's and TME's are more likely to be straight talkers who make the proper inferences about your requirements and provide you the information you need to make a decision.

4. Network with peers.

Use Twitter. Engage the Packet Pushers audience Slack channel. Try /r/networking on Reddit. Read blogs of fellow engineers you respect. Find a local NOG or MeetUp. The most honest, informative, context-correct answers you'll get are from engineers just like you who are willing to share their experiences.

5. See if your VAR (if you have one) will talk frankly.

Often, VARs are biased to push vendor solution X or Y because of partnership incentives. That's a bummer, but just as often, you can find an engineer that's willing to talk shop honestly.

If you've done a project with a VAR, you might have made friends with an engineer during the time they were on site. Often, these folks don't mind an honest chat in the back channel, even if the time isn't billable. Send them an email and offer to buy them lunch for their time and opinions.

6. Read, listen, and learn a lot.

The more you know about how technology in a general sense is supposed to work, the more you know about how to interpret a vendor's implementation of that technology.

So learn the standards and protocols, as well as the problems these standards and protocols were created to address. Then you have context with which to discern how a vendor tackles a problem with their solution.

Sources worth considering are technical books by O'Reilly Press, Safari Books Online, the Juniper Day One series, Cisco Press, and Pearson Publishing, among others. The IETF is also a fountain of freely available technical knowledge, despite some drafts and RFCs being obtuse.

Find, where you can, vendor-independent books and whitepapers on tech and dig in. And don't be shy about vendor-specific whitepapers, by the way. While most of them are marketing drivel and not especially helpful, many of them are educational and informative.

We Packet Pushers have published nearly 1,000 podcasts across our network. You can try our Foundational Podcasts page for episodes we think are especially helpful for those on a knowledge quest.

Technical Mastery Vs. The Long View

By Tommy McNicholas

A little over two years ago I left a large enterprise to work for an integrator as a practice lead in automation. I had several years' experience with network infrastructure-as-code and private cloud design. I also had experience in multiple areas of infrastructure architecture and operations.

But I was still drowned by a virtual firehose of information when it came to automation. In my first year at the new job, Docker and Kubernetes were the hot technologies. They're rooted in software development because their real purpose is delivering software rapidly and repeatedly.

As I learned to build reliable Docker images, I thought about becoming a master practitioner. I could go back to school for computer science, notch a few more certifications, or even take up programming as a trade.

But I also wondered how those efforts might serve my overall career goals. Mastering Docker, or Kubernetes, or whatever comes next, would require one or more specializations and a lot of hands-on work with the technology.

I weighed those requirements against what I felt was my primary skill: the ability to understand multiple fields and connect technologies and people together.

Did I want to go down another tech rabbit hole, or further develop my primary skills and learn more about the wider challenges facing my customers.

Fast forward to the present: I've built a team that can handle hyper-scale infrastructure, cloud-native applications, and business service management.

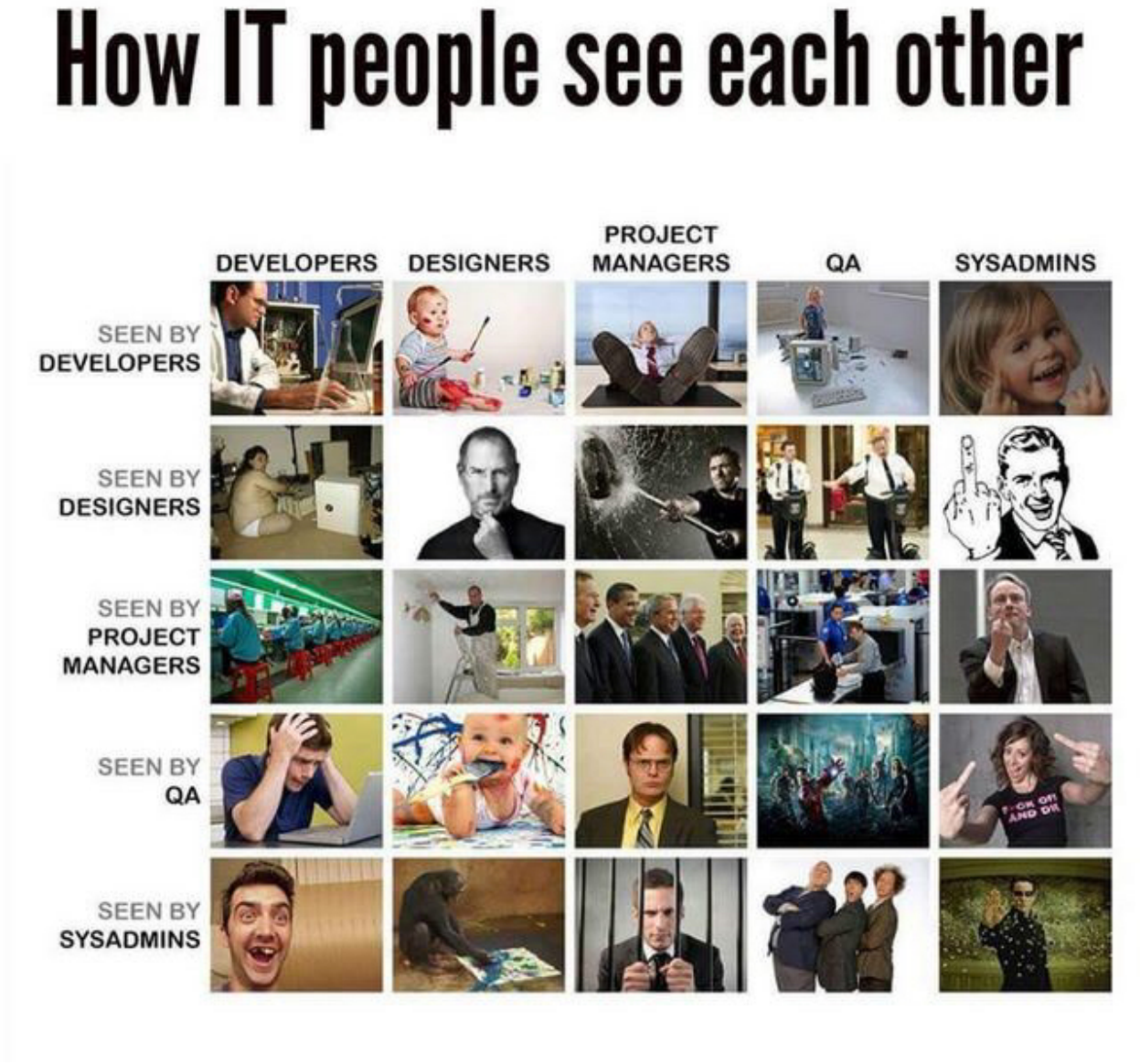
I realized that my former mode of pursuing certification and becoming an expert in a hot new tech was kind of a crutch; racking up another certifcation was familiar and comfortable, but it also distracted me from my longer-term goals.

I learn every day and I do go deep on a topic from time to time, but for the most part now I focus on understanding the principles and reasoning behind a technology and then what it takes to be successful with it.

If you love being an individual contributor and want to become a specialist in a specific technology, have at it. That choice is still an extremely valuable career move. But before you start, look at where you are in your career and make sure that this path will take you where you want to go.

Thanks Internet!

All kinds of things wash up in our social feeds. Sometimes they're worth sharing.



Internets Of Interest

Links to stories you might find interesting, curious, enraging, or otherwise worth your time.

The Disturbing High Modernism Of Silicon Valley

Cal Newport, a computer science professor and author, writes about a recently leaked Facebook memo in which an executive states that "we believe in connecting people so deeply that anything that allows us to connect more people more often is *de facto* good."

Newport critiques the notion that reason and technological innovation can remake human society to be more efficient (an idea called High Modernism) because it ignores the messy complexities of human desires, behaviors, and cultures.

And aside from the fact that Facebook has a profit motive to connect more people, even if the company took on this mission out of sheer altruism, it would still be an act of hubris with potentially bad outcomes.

"The problem with social media's attempt to improve human sociality is not the details of its implementation, it's instead the very fact that they're pursuing such a utopian

objective in the first place."

LINK

Most Important Skills In Networking

What are the most important skills in networking? Knowing CLI commands inside and out? Understanding the deep mysteries of routing protocols or spanning tree? Mastering the tools and technologies of SDN?

What about writing? Or speaking? Daniel Dib argues that these and other soft skills, including curiosity and analytical thinking, are crucial for success in networking and IT.

LINK

Recent Podcasts

The last five podcasts published on Packet Pushers

- BiB 039: Reviewing The D-Link DGS-1510-52
- BiB 038: Gluware Automates The Brownfield
- Datanauts 129: Automation And Security In AWS
- Network Break 179: Microsoft Targets Edge Computing; HCI Revenues Boom
- Show 384: The Packet Pushers Unleashed

Subscribe to the podcast on iTunes

Subscribe to the podcast via RSS

Did We Miss Something?

Got an link or an article to share? Email it to humaninfrastructure@packetpushers.net

The End Bit

Human Infrastructure is bi-weekly newsletter with view, perspectives, and opinions. It is edited and published by Greg Ferro and Drew Conry-Murray from PacketPushers.net. If you'd like to contribute, email Drew at drew.conrymurray@packetpushers.net.

We don't give away your email address or personal details because that would suck.