A FEW GOOD APPS

Developer: Network team, did you order the Code upgrade?!

Operations Manager: You don’t have to answer that question!

Network Engineer: I’ll answer the question. You want answers?

Developer: I think I’m entitled!

Network Engineer: You want answers?!

Developer: I want the truth!

Network Engineer: You can’t *handle* the truth! Son, we live in a world that has VLANs, and those VLANs have to be plumbed by people with CLIs. Who’s gonna do it? You? You, Database Admin? I have a greater responsibility than you can possibly fathom. You weep for app agility and you curse the network. You have that luxury. You have the luxury of not knowing what I know, that network plumbing, while tragically complex, delivers apps. And my existence, while grotesque and incomprehensible to you, delivers apps! You don’t want the truth, because deep down in places you don’t talk about at parties, you *want* me on that CLI. You *need* me on that CLI. We use words like “routing”, “subnets”, “L4 Ports”. We use these words as the backbone of a life spent building networks. You use them as a punch line. I have neither the time nor the inclination to explain myself to a man who rises and sleeps under the blanket of infrastructure that I provide, and then questions the manner in which I provide it! I would rather you just said “thank you”, and went on your way. Otherwise, I suggest you pick up a putty session, and configure a switch. Either way, I don’t give a damn what you think you are entitled to!

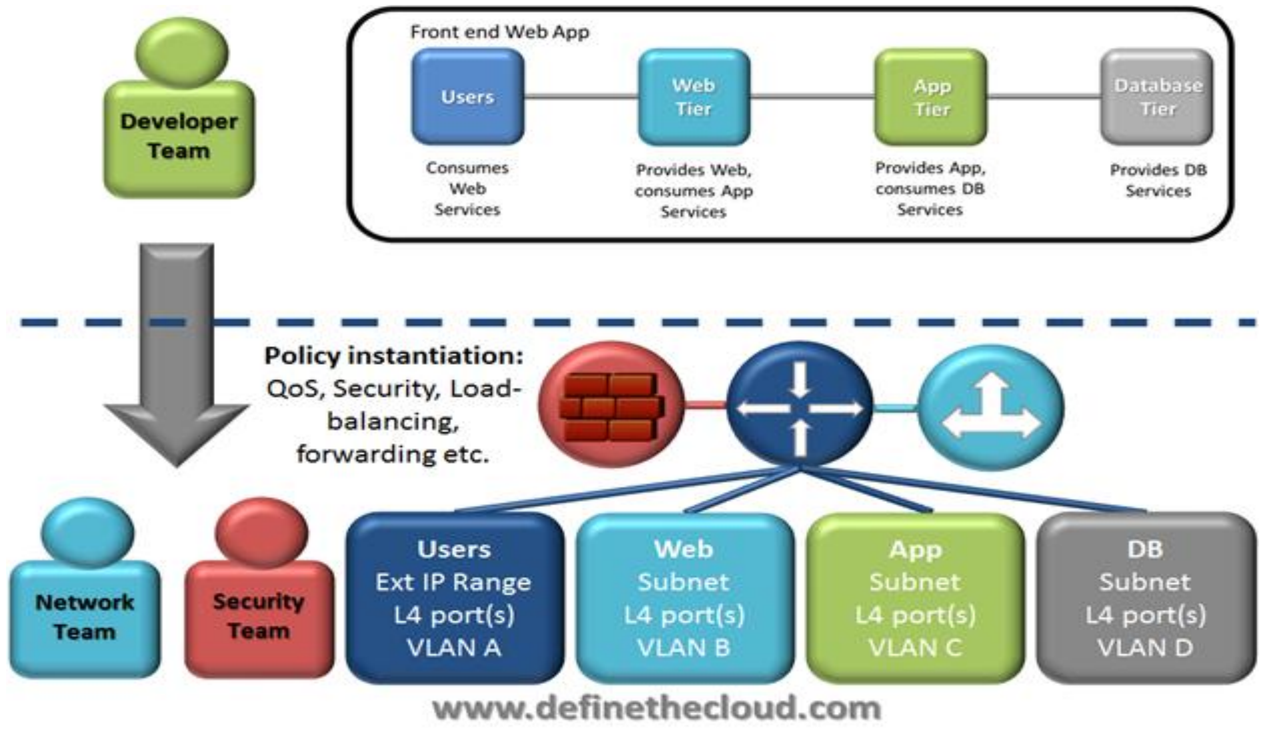
Developer: Did you order the Code upgrade?

Network Engineer: I did the job that—-

Developer: Did you order the Code upgrade?!!

Network Engineer: YOU’RE GODDAMN RIGHT I DID!!

In many IT environments today there is a distinct line between the application developers/owners and the infrastructure teams that are responsible for deploying those applications. These organizational silos lead to tension, lack of agility and other issues. Much of this is caused by the translation between these teams. Application teams speak in terms like: objects, attributes, provider, consumer, etc. Infrastructure teams speak in memory, CPU, VLAN, subnets, ports. This is exacerbated when delivering apps over the network, which requires connectivity, security, load-balancing etc. On today’s network devices (virtual or physical) the application must be identified based on Layer 3 addressing and L4 information. This means the app team must be able to describe components or tiers of an app in those terms (which are foreign to them.) This slows down the deployment of applications and induces problems with tight controls, security, etc. I’ve tried to describe this in the graphic below (for people who don't read good and want to learn to do networking things good too.)

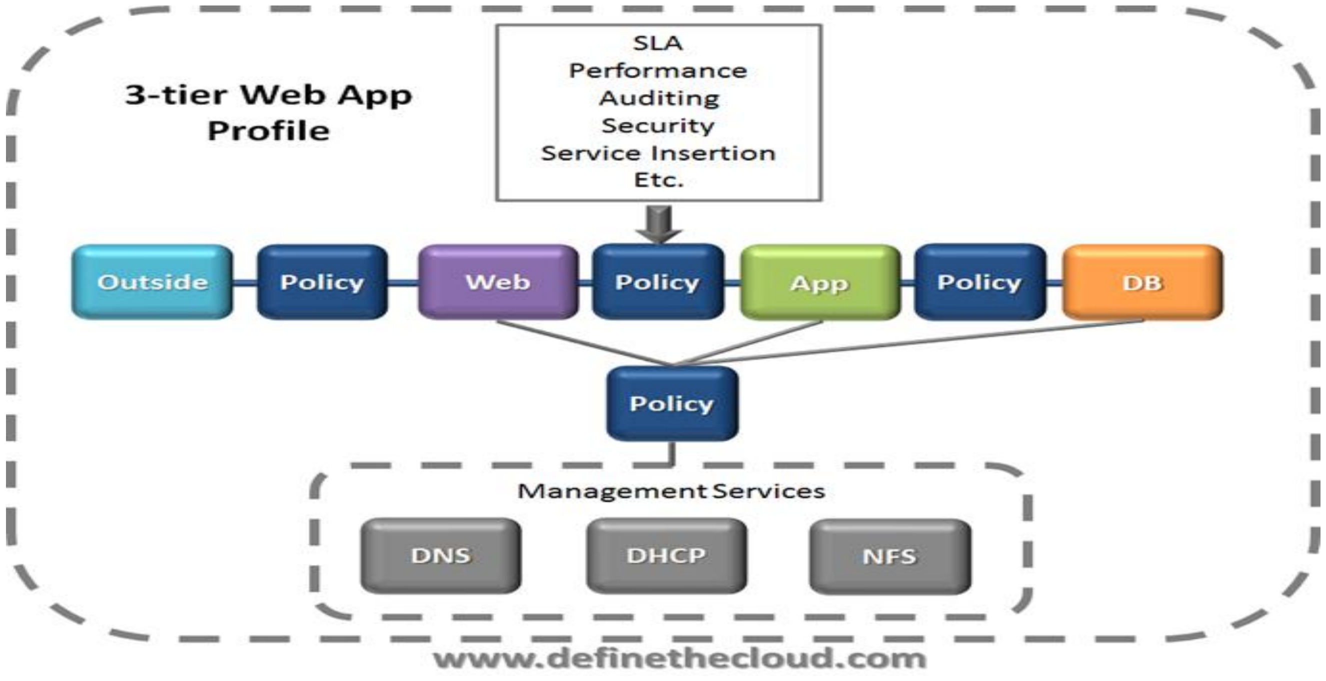

Developer Team
Front end Web App
Users
Consumes Web Services
Web Tier
Provides Web, consumes App Services
App Tier
Provides App, consumes DB Services
Database Tier
Provides DB Services
Policy instantiation: QoS, Security, Load-balancing, forwarding etc.
Network Team
Security Team
Users Ext IP Range L4 port(s) VLAN A
Web Subnet L4 port(s) VLAN B
App Subnet L4 port(s) VLAN C
DB Subnet L4 port(s) VLAN D
www.definethecloud.com

As shown in the graphic, the definition of an application and its actual instantiation onto networking devices (virtual and physical) is very different. This causes a great deal of the slowed application adoption and the complexity of networking. Today's networks don't have an application centric methodology for describing applications and their requirements. The same can be said for emerging SDN solutions. The two most common examples of SDN today are OpenFlow and Network Virtualization. OpenFlow simply attempts to centralize a control plane that was designed to be distributed for scale and flexibility. In doing so it uses 5-tuple matches of IP and TCP/UDP headers to attempt to identify applications as network flows. This is no different from the model in use today. Network virtualization faithfully replicates today’s network constructs into a hypervisor, shifting management and adding software layers without solving any of the underlying problem.

What’s needed is a common language for the infrastructure teams and development teams to use. that common language can be used to describe application connectivity and policy requirements in a way that makes sense to separate parts of the organization and business. Cisco Application Centric Infrastructure (ACI) uses policy as this common language, and deploys the logical definition of policy onto the network automatically.

Cisco ACI bases network provisioning on the application and the two things required for application delivery: connectivity and policy. By connectivity we’re describing what group of objects is allowed to connect to other groups of objects. We are not defining forwarding, as forwarding is handled separately using proven methods, in this case ISIS with a distributed control plane. When we describe connectivity we simply mean allowing the connection. Policy is a broader term, and very important to the discussion. Policy is all of the requirements for an application: SLAs, QoS, Security, L4-7 services etc. Policy within ACI is designed using reusable ‘contracts.’ This way policy can be designed in advance by the experts and architects with that skill set and then reused whenever required for a new application roll-out.

Applications are deployed on the ACI fabric using an Application Network Profile. An application network profile is simply a logical template for the design and deployment of an applications end-to-end connectivity and policy requirements. If you're familiar with Cisco UCS it's a very similar concept to the UCS Service Profile. One of the biggest benefits of an Application Network profile is its portability. They can be built through the API, or GUI, downloaded from Cisco Developer Network (CDN) or the ACI Github community, or provided by the application vendor itself. They're simply an XML or JSON representation of the end-to-end requirements for delivering an application. The graphic below shows an application network profile.



This model provides that common language that can be used by developer teams and operations/infrastructure teams. To tie this back to the tongue-in-cheek start to this post based on dialogue from “A Few Good Men”, we don’t want to replace the network engineer, but we do want to get them off of the CLI. Rather than hacking away at repeatable tasks on the command line, we want them using the policy model to define the policy ‘contracts’ for use when deploying applications. At the same time we want to give them better visibility into what the application requires and what it’s doing on the network. Rather than troubleshooting devices and flows, why not look at application health? Rather than manually configuring QoS based on devices, why not set it per application or tier? Rather than focusing on VLANs and subnets as policy boundaries why not abstract that and group things based on those policy requirements? Think about it, why should every aspect of a servers policy change because you changed the IP? That’s what happens on today’s networks.

Call it a DevOps tool, call it automation, call it what you will, ACI looks to use the language of applications to provision the network dynamically and automatically. Rather than simply providing better management tools for 15 year old concepts that have been overloaded we focus on a new model: application connectivity and policy.

Source: http://www.definethecloud.net/a-few-good-apps/