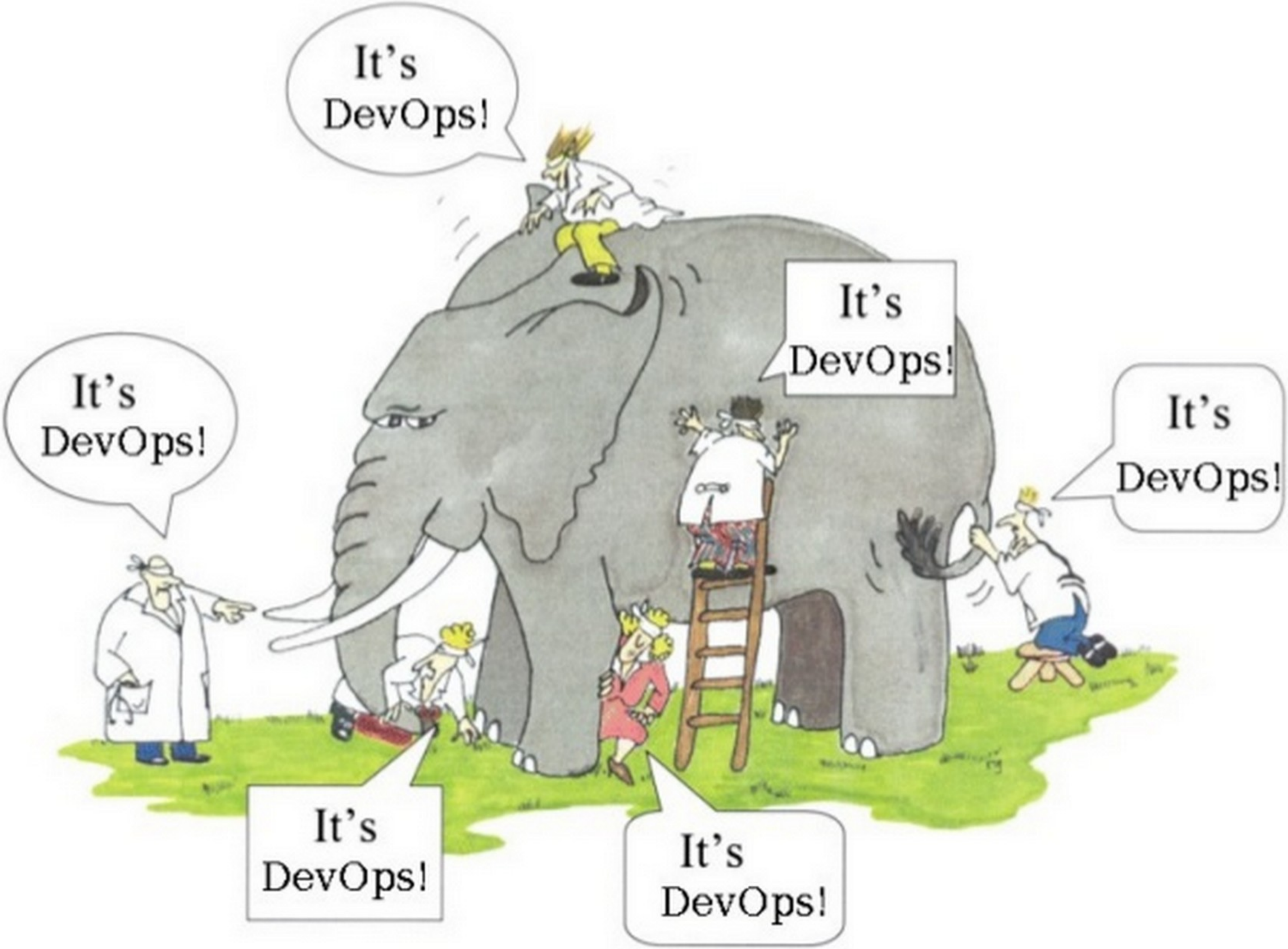
Introduction to DevOps Technologies

YOU **FOOL**!

It’s not about the tools, it’s about the **CULTURE**!


It's DevOps!
It's DevOps!
It's DevOps!
It's DevOps!
It's DevOps!
It's DevOps!

What is DevOps?

“ DevOps is a software development method that stresses Communication, Collaboration, Integration, Automation Measurement of cooperation between software developers and other information-technology (IT) professionals."

- Wikipedia

What is Operations ?

Operations is simply the set of processes and services provisioned by IT personnel to their own internal or external clients in order to run their Business.

It is generally delineated in several ways:

- Infrastructure and Monitoring
- Architecture and Planning
- Maintenance
- Support

What is Development ?

Its a process of programming, documenting, testing and debugging associated with application development and the release life cycle.

Methodologies used for software development

- Prototyping
- Waterfall
- Agile
- Rapid

Silos of Responsibility

When did that begin to change? Well, probably when things got a little bit “cloudy”.

Breaking Down Barriers

- In about 2008, when Agile software development began to gain steam as a methodology, the concept of DevOps was introduced.

- Around 2010 as Amazon’s relatively new “Internet data center” became more popular, the skill sets of both of these silos began to converge.

- Now that anyone could provision “images” to use for rapid development and prototyping, the skills necessary to manage those configurations started to be more well understood in general.

- These “crossover” skills began to create a new type of engineer that was exactly as described and those barriers began to break down.

DevOps Culture

- Companies are using the DevOps rallying cry as an excuse for not hiring the staff needed.

- DevOps allows companies rapid development lifecycles

- Your ability to deploy new features on your site or in your application in a constant manner can be the difference between succeeding or failing in the market.

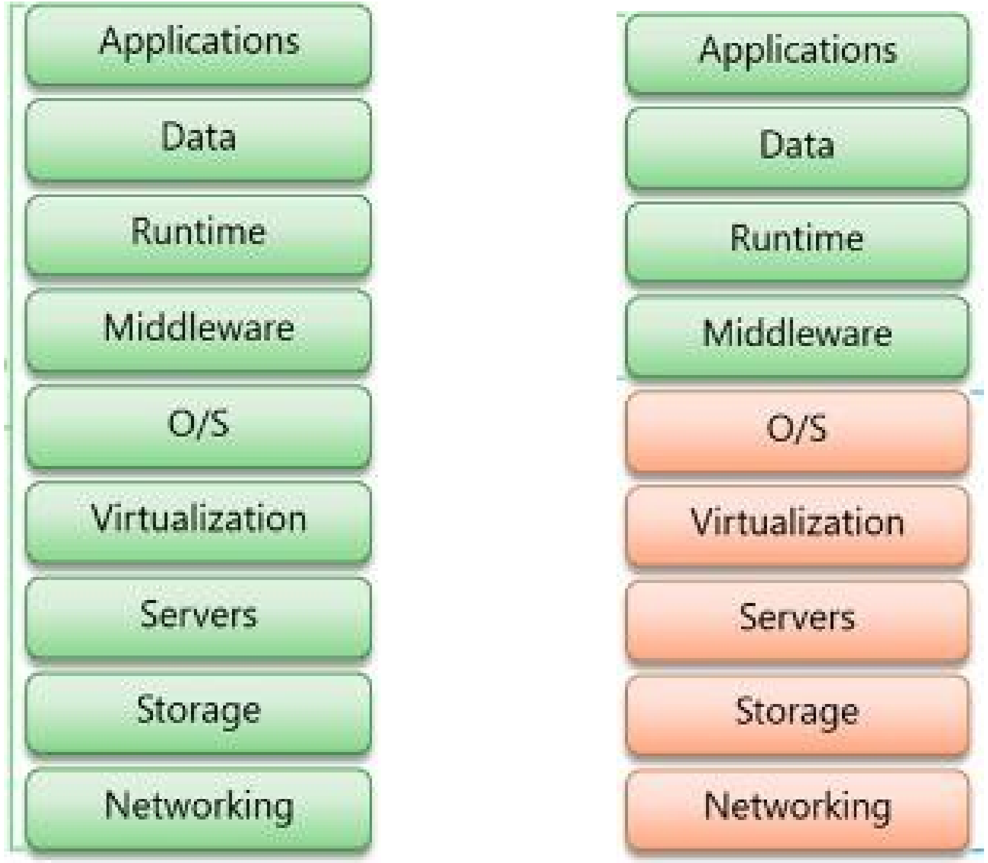
Terminology: IaaS?

- One of the first steps in the DevOps revolution was when Information Technology began to look at it’s Infrastructure differently.

- IT in general, operations specifically, was always seen as a “cost center”. A necessary budgetary evil that every company had to accept but was not well understood or appreciated.

- Smart executives in IT began to change the model of how Infrastructure was consumed. It started with charging Infrastructure costs back to the business units that consumed them. In short, IT began Infrastructure as a Service.

Traditional Services vs IaaS View

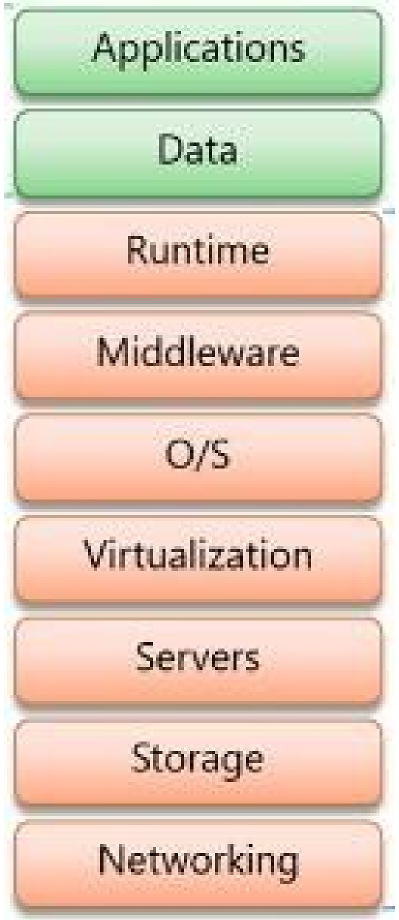


Influence of IaaS on Devops

It certainly has influenced the services and skills that DevOps requires.

As virtualization has overtaken dedicated hardware resources in particular, IT has had to develop different (and faster) skills around the scale and speed of traditional infrastructure deployments.

Terminology: PaaS?



In the Platform as a Service model, IT (or a vendor/cloud provider) delivers a “computing platform” for consumption.

It generally includes everything from the previously detailed IaaS model as well as a few additions.

Sample platforms are database, web servers, runtimes,etc. These are independent of the Infrastructure but pushes the service offering up and takes more of the burden off the business or the developers to manage.

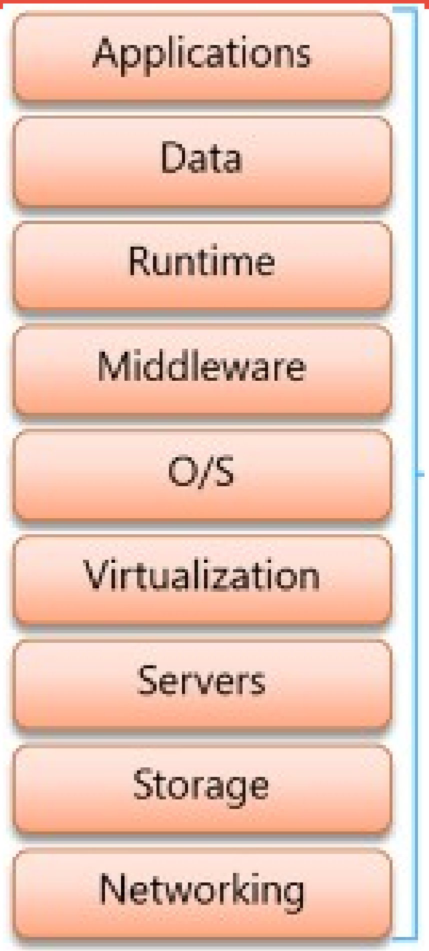
Why Is PaaS Important?

Again, although not the driving force behind the DevOps movement, it had significant influence as a further evolution of how IT was traditionally thought of.

Big name vendors like Microsoft (Azure) and Google (App Engine) were some of the first to offer the underlying computer and storage resources that could scale automatically to match application demand so manual allocation of resources was no longer necessary.

It also converged developer and operations skill sets even more than before.

Terminology: SaaS

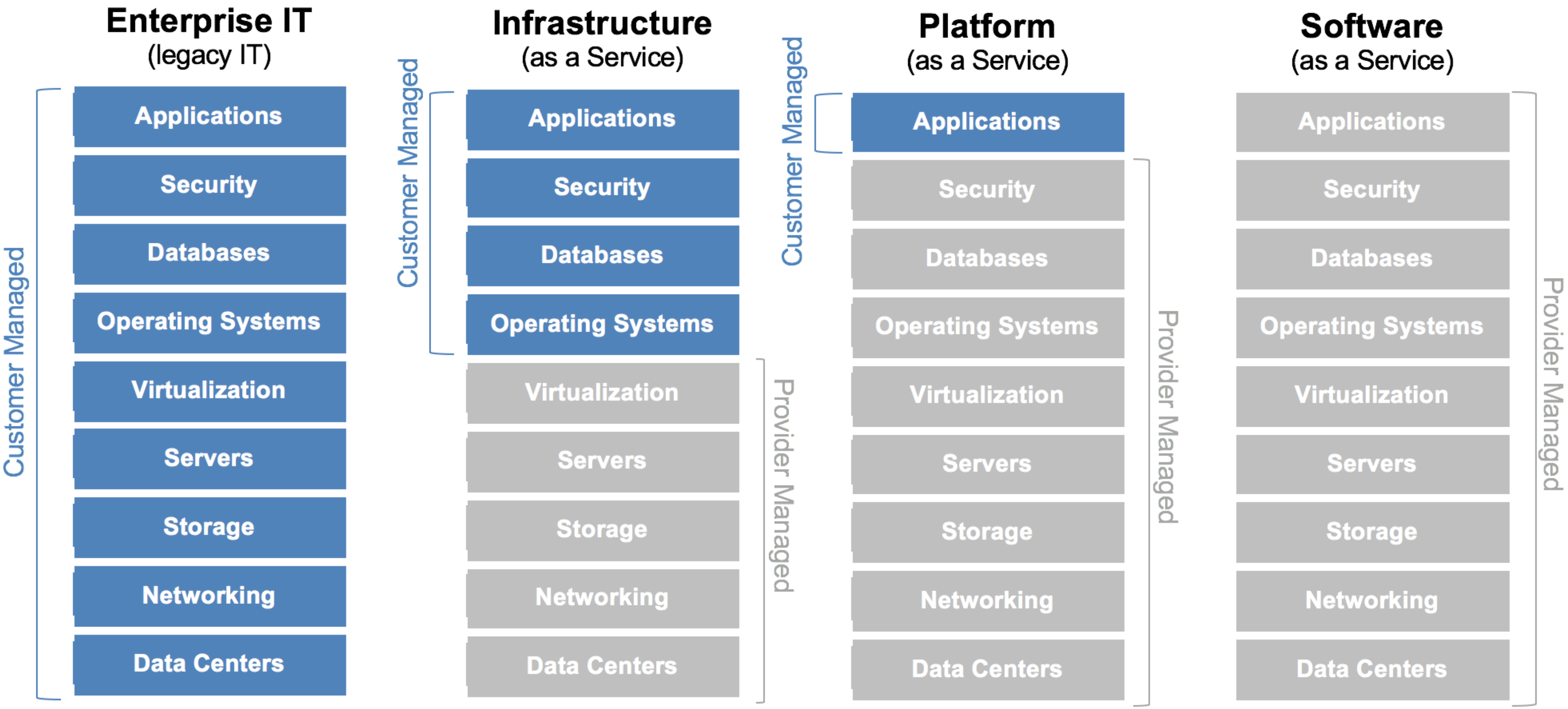


In the Software as a Service model, IT (or a vendor/cloud provider) delivers ACCESS to the software to be used without having to do anything to manage, configure, monitor or support it.

This eliminates the need to install or run applications locally and can be run entirely remotely (“in the cloud” so to speak).

Since everything in the traditional stack is now consumed as a service, those lines, well, they disappear.

Summary of Cloud technologies



Enterprise IT
(legacy IT)

Customer Managed

Applications
Security
Databases
Operating Systems
Virtualization
Servers
Storage
Networking
Data Centers

Infrastructure
(as a Service)

Customer Managed

Applications
Security
Databases
Operating Systems

Provider Managed

Virtualization
Servers
Storage
Networking
Data Centers

Platform
(as a Service)

Customer Managed

Applications

Provider Managed

Security
Databases
Operating Systems
Virtualization
Servers
Storage
Networking
Data Centers

Software
(as a Service)

Provider Managed

Applications
Security
Databases
Operating Systems
Virtualization
Servers
Storage
Networking
Data Centers

People and skills required

The skills necessary to manage these resources now cross over these previously traditional silos.

No longer do we have the software developer who doesn’t understand basic networking or the hardware engineer who cannot develop, they are required skills and contained in the same space.

Build Automation - Basics

The process of “building” or compiling software that can then be deployed via script or cron jobs to various environments, including production systems.

In the DevOps world, it encompasses not only the software portion, but the process of automating the deployment of compute resources (physical or virtual, applications and data).

Build automation in DevOps terms allows the deployment and management of the entire stack of services, without manual intervention.

Treat Your Infrastructure As Code

DevOps erases those traditional lines between Operations and Development.

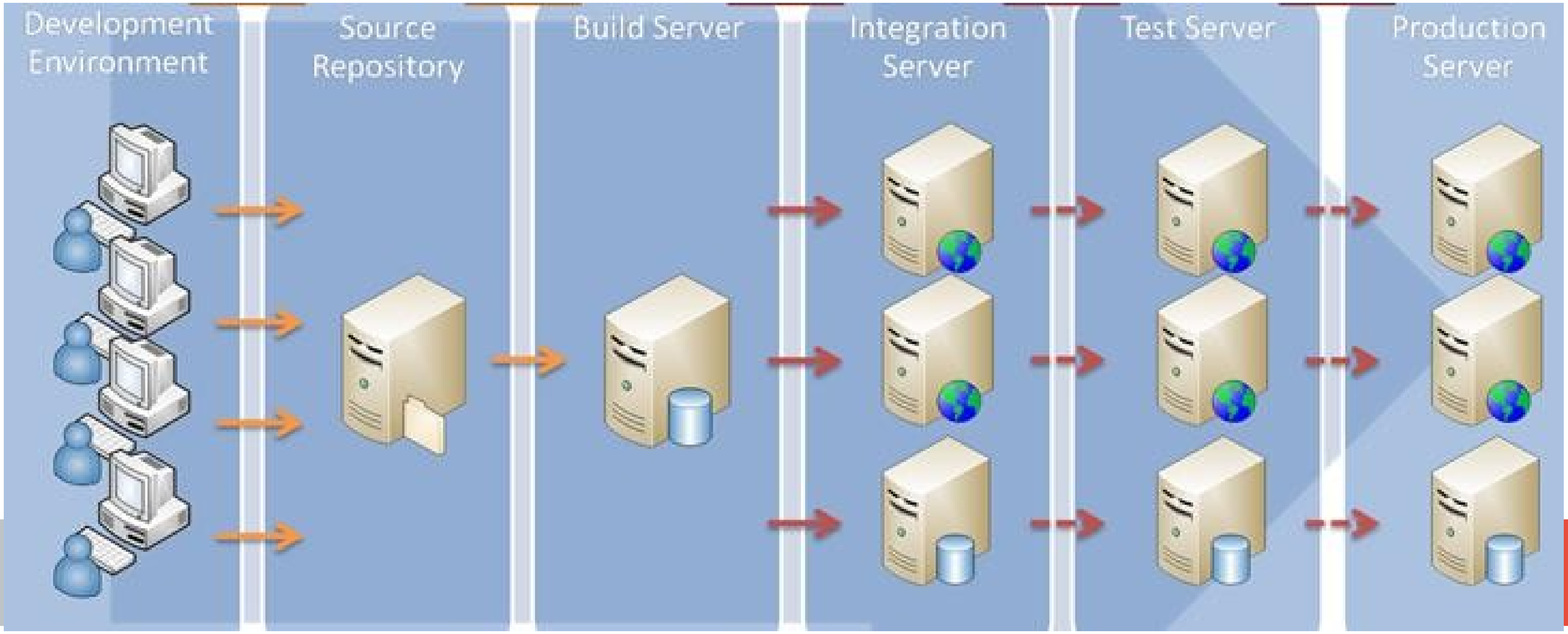
It does so because everything is treated as a “compute resource” and can be managed with code.

When your compute resources are largely virtual (cloud consumed), your deployments can be automated throughout the stack with build automation tools.

What Does It Look Like?

Build automation is the process by which you initiate a software or hardware deployment automatically, using consistent methods, all the way through the environment stack.

It can and does include automated testing and rollback capabilities so that each environment remains stable and consistent.



Definition: Continuous Integration

This can be defined as the practice of merging development working copies with the shared source main (branch) multiple times per day.

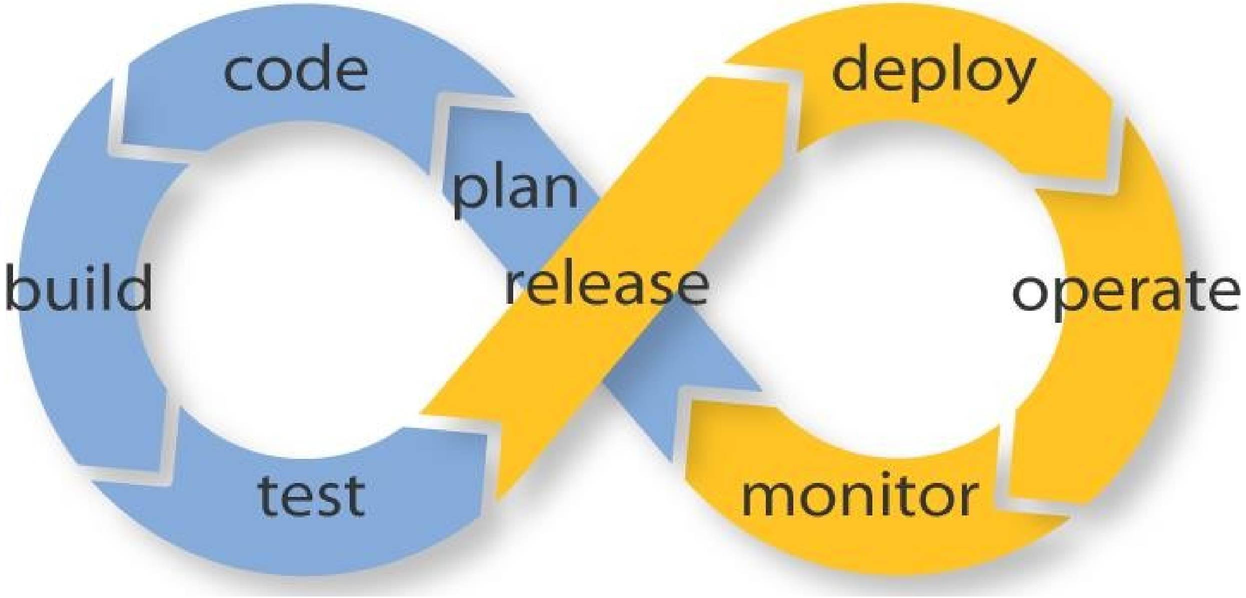
The concept of multiple integrations per day on the main source branch is to prevent integration problems in large development teams where the odds of one change breaking the changes of another developer would be smaller.

It takes advantage of those automated build processes to build and test each commit and reporting those results back to the development teams.

Definition: Continuous Delivery

This is an approach in which software teams keep producing valuable software in very short delivery cycles and ensures that those features can be reliably and consistently released at any point in time.

What Does It Look Like?



CI and CD are designed to take advantage of taking things in smaller chunks to increase stability and decrease release cycles.

As one progresses constantly, the outputs from the CI process feed into the CD process and the cycle begins again.

DevOps is driven largely by these two processes as they are the culmination of treating everything like code in rapid cycles.

Next : Tools used in DevOps

Jenkins

In short, Jenkins is Build Automation on steroids. It also can be considered a Continuous Integration tool. Often considered a Continuous Deployment tool as well.

Although used more often in software development for build deployments, it supports a large number of plugins that enable anything from deploying scripts to launch virtual machines through VMWare or Vagrant to Docker containers across environments.

Was originally called Hudson in 2004 and started at Sun Microsystems. After big bad Oracle bought Sun and claimed the name, it was changed to Jenkins in 2011.

What Does It Look Like?

Jenkins allows you to create build jobs that do anything from deploying a simple software build to the custom creation of a Docker container with specific build branches while doing performance and unit testing while reporting results back to the team

The variety of plugins you can download and install as well as the way you can manage and distribute the build load across multiple slave servers ensures that you have a robust and performant environment for automating all facets of your build, integration and deployment process.

Docker

According to Wikipedia (and others):

Docker is an open-source project that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating system-level virtualization on Linux.

So basically, it is a tool (or a set of tools depending on how you look at it) that packages up an application and all its dependencies in a “virtual container” so that it can be run on any Linux system or distribution.

When Would I Use Docker?

There are a lot of reasons to use Docker. Although you will generally hear about Docker used in conjunction with development and deployment of applications, there are a ton of examples for use:

- **Configuration Simplification**
- **Enhance Developer Productivity**
- **Server Consolidation and Management**
- **Application Isolation**
- **Rapid Deployment**
- **Build Management**

Is Containers new concepts ?

Containers are not a new concept in technology, it just appears that Docker has captured the buzz (right place, right time). If you look around, you will find that a number of companies and projects have been working on the concept of application virtualization for some time:

- FreeBSD – Jails
- Sun (and now Oracle) Solaris – Zones
- Google – lmctfy (Let Me Contain That For Your)
- LXC (Linux Containers)

What is Chef?

- It is a configuration management tool written in the Ruby and Erlang software languages.
- You create “recipes” using pure-Ruby (a domain specific language, or DSL).
- Chef is most often used to streamline configuration tasks for maintaining servers and related infrastructure.
- It easily integrates with cloud-based platforms (AWS, Google Compute, Azure, OpenStack, etc) to provision and configure new “machines”, “images” or “instances” automatically.
- It can also be used to configure almost anything that runs ON those instances as well.

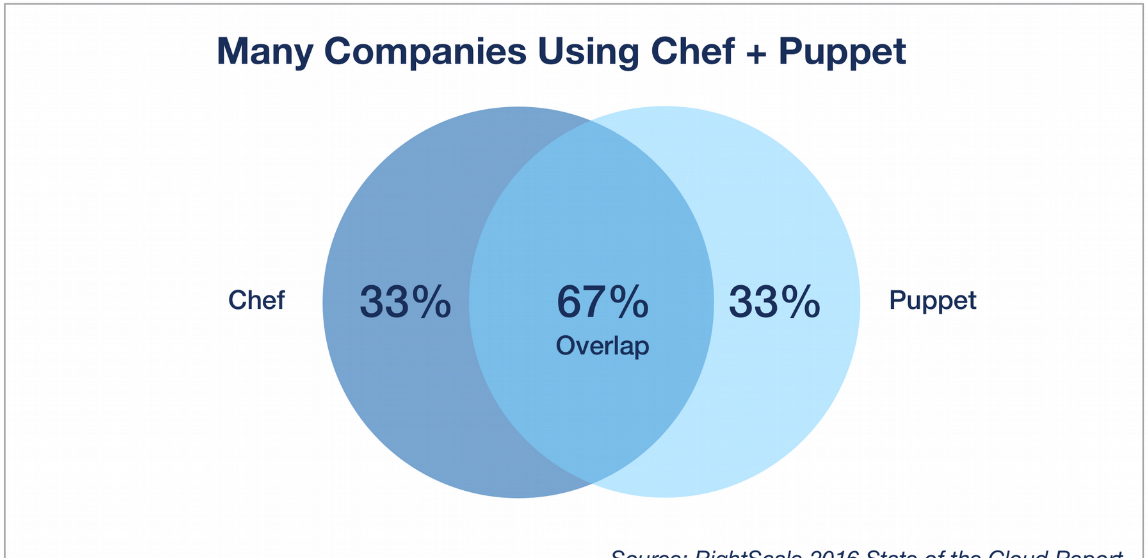
What is Puppet?

Puppet is also a configuration management tool or utility that runs on Unix/Linux/Windows/OSX.

It includes a custom “declarative” language that is used to “describe” system configurations and their state.

Although it uses its own declarative language, like Chef, you can also use a Ruby DSL (domain specific language).

All of your configuration scripts are stored in files referred to as “Puppet Manifests”.



Many Companies Using Chef + Puppet

Source: RightScale 2016 State of the Cloud Report

Why are chef and puppet important to DevOps?

These tools allow you to live one of the DevOps rallying cries – “treat your infrastructure as code”

By abstracting the configuration needed to launch server instances with specific configurations into consistently repeatable “recipes” or “manifests”, the skill sets needed to manage the entire technology stack is converged – both operations and development skills are required.

Thank You !