RESEARCH ARTICLE OPEN ACCESS

Accelerating product time to market using PaaS and Automation tool

Swara Pampatwar [1], Ritu Godhani [2], Manisha Ghorpade [3]

[1]Department of Computer Science, Jhulelal Institute of Technology., Nagpur, INDIA
Email: padmajaadgulwar@gmail.com
[2]Department of Computer Science, RCOEM, Nagpur INDIA
Email: rbg412@gmail.com
[3]Department of Computer Science, RCOEM, Nagpur INDIA
Email: manishaghorpade123@gmail.com

ABSTRACT
In today's product development environment, product and the technologies they're based on change rapidly, as do the number of competitors for market share. This means that time to market and finding ways of optimizing it are critical components that directly affect revenue. Organizations always make a systematic approach to reduce time to market. However, in today's world of Big Data and cloud computing, there are some complex situations Because of which product development gets delayed. Thus, it becomes necessary to have suitable automation tool for a speedy development of the product development gets delayed. Thus, it becomes necessary to have suitable automation tool for a speedy development of the product.

Keywords **–** PaaS, Puppet, Time to market.

I. INTRODUCTION

In Software Development, it is perceived that having an idea is only half the battle, but in age of big data and Cloud to create a successful, fully realized product, you also need multiple teams working together in synchronously. For this, Need of PaaS and Suitable Automation come into the Picture.

II. NEED OF PAAS FOR SPEEDY PRODUCT DEVELOPMENT:

For development team to kick-start with development of product, they need working environment which takes lot of time sometimes in days for environment to get ready. This affects the deadline so there should be some way which could prepare machines/Environments on fly for various teams and PaaS is capable of doing this.

Platform as a service (PaaS) is a category of cloud computing services that provides a computing platform and a solution stack as a service. Along with software as a service (SaaS) and infrastructure as a service (IaaS), it is a service model of cloud computing. In this model, the consumer creates the software using tools and/or libraries from the provider. The consumer also controls software deployment and configuration settings. The provider provides the networks, servers, storage, and other services.

In a PaaS model, the vendor not only provides the underlying infrastructure resources but also the application development platform. This platform includes the automation to deploy, test and iterate applications. Operating systems, databases, middle-ware and up-to-date tools and services are all provided by the PaaS vendor, so time-consuming operational tasks such as configuring, optimizing and continuously updating your environment are handled on your behalf.

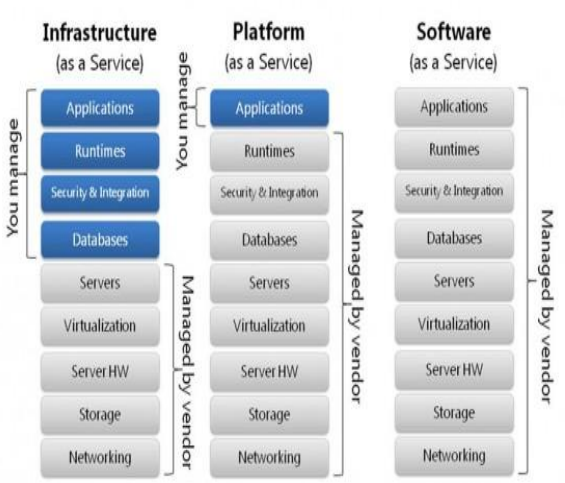

Fig 1: Comparison between IaaS, PaaS and SaaS

III. NEED OF AUTOMATION TOOL

Cloud and virtualization are accelerating the demand for automation. Why? Because running jobs and processes across diverse machines and infrastructure requires resources. Generally, it is

considered a good practice to take a configuration backup before making any changes to the production configuration so that the old configuration is available for roll-back. Also, as a part of the organization's policy, the same base configuration should be configured on all the servers to reflect consistency and as a server-hardening practice. A majority of the problems in the UNIX environment occur due to ad-hoc changes, which can be mitigated by following proper change management procedures. Handling and monitoring ad-hoc changes, and restoring the previous state, remains a challenge for organizations. Meeting such challenges is quite workable for a small set-up of 1-20 servers and a dedicated UNIX administration. But during hardware failure or other problems, where the servers need to be reconfigured from scratch, it takes a lot of effort and time in restoring the servers to the previous state.

To handle such scenarios, a quick solution would be to hire another UNIX administrator who could act as a secondary resource and offloads other activities from the primary resource during disaster conditions. Think about a scenario of managing a globally-distributed data centre with 500 UNIX servers or more, comprising Solaris, Debian, Ubuntu, Fedora, CentOS, etc. Here, servers are running with the same base configuration and packages, where configuration files need to be checked-out to a version-controlled repository. Only planned changes are allowed and the previous configuration state is restored for unplanned changes. Additionally, centralized user and policy management, along with automated configuration recovery during disaster conditions are required. In such a case, building a team of 10-20 administrators would not be a recommended approach. Rather, using a centralized configuration tool to automate the administration tasks would be a better option to follow. **[1]** The products like CFEngine, Puppet and Chef are all seen as configuration management solution, they differ in their approach. They all have unique architecture and decision making process that in-turn applies configuration polices system wide. They all are capable of managing and configuring all the aspects of the system and quite successful in their field compared to other products. They all have taking a different route to address the problem of system administration and automation. [5]

3.1 PUPPET

So here comes need of automation and configuration management tools available like Chef and Puppet. Puppet turns out to be a next-generation configuration management tool. Puppet is IT automation software that helps system administrators manage infrastructure throughout its lifecycle, from provisioning and configuration to orchestration and reporting. Using Puppet, you can easily automate repetitive tasks, quickly deploy critical applications, and proactively manage change, scaling from 10s of servers to 1000s, on-premise or in the cloud

Puppet is written in Ruby and is released under the GPL. It supports a number of operating systems like CentOS, Debian, FreeBSD, Gentoo, OpenBSD, Solaris, SuSE Linux, Ubuntu, etc. Puppet installation is fairly easy and is, in fact, a matter of seconds. Puppet runs in client-server configuration, where the client polls the server at port 8140 every 30 minutes to check for the new instructions or to match the configuration files. The client also listens to a port to have push-updates from the server. In Puppet terminology, a client is called a Puppet node and a server is called a Puppet master. [1]

While virtualization allows for greater service isolation and hardware utilization, each virtual machine represents as much configuration as a physical machine. With 10 or more virtual machines running on the same hardware or a cluster of VMs started with Web services, IT organizations can reduce hardware and capital expenses, but they multiply the configurations that need to be managed Image-based service management seems like a solution. The problem with the image-based approach is that after months of managing images, sprawling configuration drift with a sprawling collection of machine images with little or no insight into each one.

With a configuration management tool like Puppet, the notion of an API extends to the configuration of each system. The encoded services provide not only a mechanism for building and maintaining systems, but the code can also provides something a 500 MB machine image doesn't match semantics. The code provides insight into not only what is configured on a system, but why. Understanding why allows for much better decision making for managing existing systems and designing new services.

Once the configuration of the Web server, application server or database server is defined and maintained by Puppet, adding a new server is almost trivial. [3]

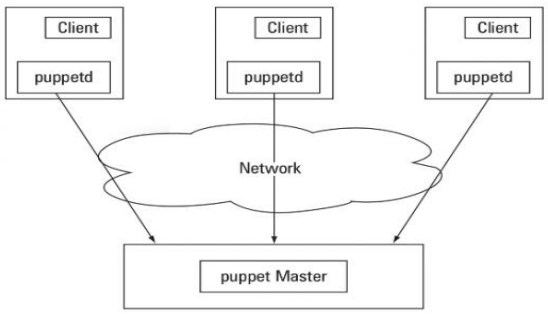

Fig 2: A typical Puppet Setup

3.2 HOW PUPPET WORKS

Puppet uses a declarative, model-based approach to IT automation.

1. **Define** the desired state of the infrastructure's configuration using Puppet's declarative configuration language.
2. **Simulate** configuration changes before enforcing them.
3. **Enforce** the deployed desired state automatically, correcting any configuration drift.
4. **Report** on the differences between actual and desired states and any changes made enforcing the desired state. [4]

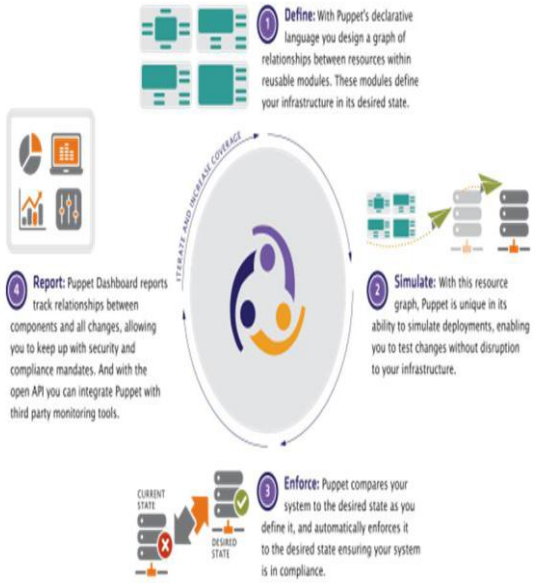

Fig 3: How puppet Works

3.2.1 Define Reusable Configuration Modules

To define infrastructure's desired state, select pre-built, freely downloadable configuration modules in the Puppet Forge, Puppet Labs' online marketplace. [3] Alternatively, build a custom module using Puppet's configuration language. Once defined, reuse these configurations across physical, virtual, and cloud environments as well as across operating systems.

These Configuration modules are combined to create complete application configuration stacks that share common configurations. With Puppet declarative language, a graph of relationship between resources within reusable modules is developed.

Fig 4: Configuration Modules

3.2.2 Enforce Desired State

After deploying configuration modules, the Puppet Agent on each node communicates regularly with the Puppet Master server to automatically enforce the desired states of the nodes.

1. The Puppet Agent on the node sends Facts or data about its state, to the Puppet Master server.
2. Using the Facts, the Puppet Master server compiles a Catalog, or detailed data about how the node should be configured, and sends this back to the Puppet Agent.
3. After making any changes to return to the desired state (or, in "no-op mode," simply simulating these changes), the Puppet Agent sends a complete Report back to the Puppet Master.
4. The Reports are fully accessible via open APIs for integration with other IT systems.

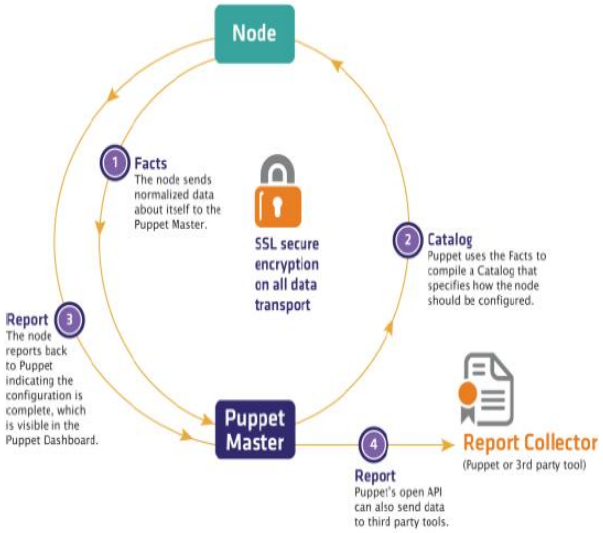

Fig 5: Communication between Puppet Master and Puppet Agents

IV. ADVANTAGES

By decreasing the development time needed before launching a product, companies are able to decrease their investment costs and realize revenue faster.

V. CONCLUSION

Thus, this paper signifies the importance of PaaS and Automation tool like puppet and when they used wisely will ensure that Time to market is reduced. Thus, there will be faster launch, Lower cost and higher market share.

REFERENCE

[1] Puppet show automating UNIX administration,"http://www.linuxforu.com/2009/06/puppet-show-Automating-Unix-administration."

[2] Platform as a service (Paas) vs. Infrastructure as a service (IaaS), "https://www.engineyard.com/paasvs-iaas."

[3] What is the puppet configuration management tool, and how does it work? "http://searchdatacenter.techtarget.com/tip/What-is-the-Puppet-configuration-management-tool-and-how-does-it-work"

[4] What is Puppet? "http://puppetlabs.com/puppet/what-is-puppet".

[5] "Investigating Community, Reliability and Usability of CFEngine, Chef and Puppet"By Sudhir Pandey Network and System Administration Oslo and Akershus University College May 22, 2012