A Survey of DevOps tools for Networking

Jay Shah
Telecommunication and Network Engineering
Southern Methodist University
Dallas, Texas. USA
jayashoks@smu.edu

Dushyant Dubaria
Telecommunication and Network Engineering
Southern Methodist University
Dallas, Texas. USA
ddubaria@smu.edu

Prof. John Widhalm
Telecommunication and Network Engineering
Southern Methodist University
Dallas, Texas. USA
jwidhalm@lyle.smu.edu

Abstract— ***DevOps is a new methodology that combines developers and operations team to closely integrate people, processes and technology for an automated software delivery that is agile, scalable and cost-effective. DevOps delivery is a combination of two previous methods; agile software development and the collaboration between developments and operations team by changing the way we think to deploy an infrastructure through the entire product life-cycle, from the design through the production phase, we can deliver consistent and repeatable designs. By abstracting the configuration needed to launch server instances with specific configurations into consistently repeatable recipes or manifests, the entire technology stack becomes converged. In order to do this, both operation and development skills are required. We have GIT as source control tool, Ansible as configuration management, Jenkins as configuration integration, Vagrant as provisioning and Docker as containerization tool. With all these tools we can build an entire Continuous Integration and Continuous Delivery pipeline.***

Keywords—**DevOps, Networking, Vagrant, Docker, Jenkins, Ansible, Git, Pipeline, CI/CD**

I. INTRODUCTION

DevOps evokes an agile software development approach in an operational environment [1]. Deployment is the period between the completion of the code by the developers and then placing of the code into normal production. Modern development organizations require entire teams of DevOps to automate and reduce the gap between the development team and the operation team [14]. This developer/operations collaboration enables to manage the complexity of real-world problems. This paper layout the basic concepts, reviews and surveys of DevOps as a whole, philosophy, workflow, monitoring methods etc. DevOps uses tools such as Git, Vagrant, Docker, Ansible, Jenkins [3]. We have integrated all the above tools to work together to host an application-based web server. And to test all our we have built a CI/CD pipeline.

II. BACKGROUND

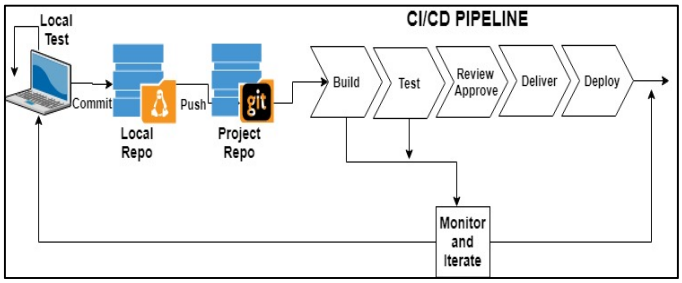

Fig. 1: DevOps Culture of Collaboration

DevOps acknowledges the interdependence of software development, quality assurance, and IT operations, and intends to help an organization rapidly produce software products and services and to improve operations performance [15] [16]. Operations are simply the set of processes and services provisioned by IT personnel to meet the requirements of their own internal or external clients in support of the business objectives. It is delineated in several ways: Infrastructure and Monitoring, Architecture and Planning, Maintenance and Support Although not necessarily exclusively, in IT, development refers to the process of creating software. It involves the programming, documenting, testing and debugging software with application development and the associated software release lifecycle. There are several methodologies for doing: Prototyping, Waterfall, Agile and Rapid [1]. Implementing DevOps also means measuring and monitoring efficiently. As a result, having effective monitoring is crucial across all components of infrastructure and networking because various monitoring points can be aggregated and monitored as real-time feedback which results in a much more rapid root cause analysis [14]. Having an effective monitoring solution helps to facilitate the metric of mean time to resolve issues. As a result, when a production issue occurs, the source of the problem can be isolated and resolved in near-real time fashion [12]. This automated approach to issue resolution has obvious benefits in terms of ensuring minimal disruption of the business objectives.

III. DEVOPS TOOLS

A. VAGRANT

Virtualization is not a new technology. It has the roots as far as back the early 1970s when the first IBM solution for Time Sharing was introduced [5]. Virtualization simply refers to abstracting the hardware components of a physical machine so that it can operate multiple virtual machines sharing local resources.

Vagrant is one of the DevOps tools that help automate infrastructure provisioning through the use of various providers which include VirtualBox, VMware, Hyper-V and AWS Cloud. In addition, Vagrant supports a variety of provisioners such as Ansible, Chef, Puppet, Salt, etc. Vagrant uses the command line and a text file called Vagrantfile to operate. Vagrant is configured such that a directory is created for each project [18]. Each machine will have its own directory where the Vagrant commands will be issued and directed only to the virtual machine associated with that project. In that directory, Vagrant uses the concept of boxes to provide an OS image [1]. A Vagrant box is a packaged environment like a template where the box can be created to support not only basic operating systems but also any application you wish to include. Vagrant boxes are also community driven and Vagrant cloud supports a plethora of Vagrant boxes for users to contribute [19]. Vagrant supports three modes of network access to the guest VM's: Port forwarding, Private Network and Public Network. Vagrant widely supports the variety of ways to provision the virtual machine: Shell scripting, Ansible playbooks, Chef recipes, Puppet and Salt. Docker is also a very flexible platform which supports the use of multiple providers.

B. DOCKER

Docker is the way to create Linux containers. Docker is a virtualization technology which does not rely on containers instead, it uses a Docker engine [2]. Hypervisors are used in virtual machines and Docker helps to run multiple software applications on the same machine. Each of which is run in an isolated environment called container. The software is packaged in the container and, hence, can be run on any machine that has the Docker engine installed. A container is a closed environment for the software and bundles all the files and libraries that the application needs to function correctly. Multiple containers can be deployed on the same machine and share the resources. Containers are application-centric and rely on the same physical hardware of the host machine and run the application in an isolated environment. Additionally, containers use fewer resources than virtual machines [20]. You cannot containerize a Windows or a MAC application. Docker can be installed on Windows, Linux, and MAC. But it can only run applications designed to work on a Linux kernel. Docker has two versions: the community edition (CE) which is free of charge but with community support and the enterprise edition (EE) is a paid version with enterprise-level support from Docker. Vagrant uses Vagrantfile while Docker uses Dockerfile for building containers and images. Building an image is an incremental process for example; you may download the Apache web server image (httpd) and use a Dockerfile to install components on top of it like a specific version of PHP [5]. You can add new packages, change configuration files, create new users and groups, and even copy files and directories to the image.

C. GIT

A Version Control System (VCS) is the management of changes to any kind of document but is usually used to control and manage software source code or scripts [8]. In addition, a VCS is also known as revision control or source control. There are two type of Version control system: Centralized VCS and distributed VCS. A centralized version control system is a client-server approach. The client checks out a copy from the centralized repository. All these data and history is stored and retrieved from the central repository. On the other hand, we have the distributed Version control system which is a peer to peer approach. Over copy is a repository and contains full project history [7]. Network connection is needed when it is to be shared.

Version controlling our code provides several benefits:
1) The ability to work on the same project by more than one person at the same time. Each person will have a copy of the source code to do whatever changes required, then changes are merged with the main code, after the necessary approvals.
2) The history stored by a version control system would allow a developer to examine changes to faulty code.
3) The code can be branched. Developers can work on separate branches of the source code to create and test new features. In the end, a branch can be merged with the main branch (called master).

GitHub is a repository for controlled documents which supports Git [21]. It was launched in 2008. It offers a version control repository that can be used free or for a fee, a private repository that can be used [9]. Git can be integrated with GitHub so that the repository can be accessed over the Internet from anywhere in the world instead of having to create a local repository server.

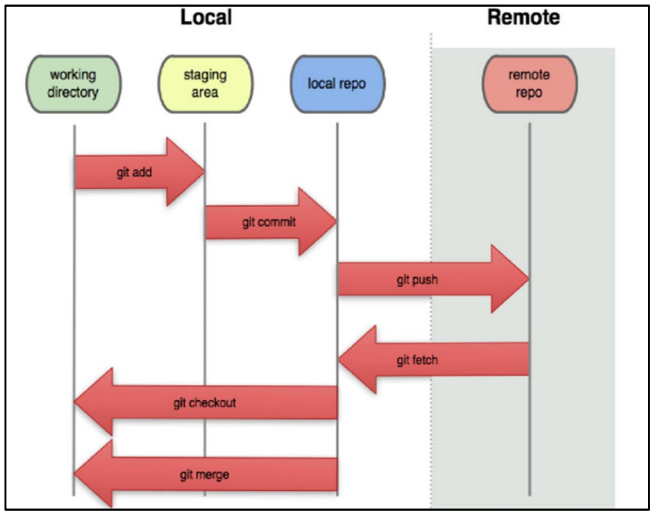

Fig. 2: GIT Working Flow

To keep a backup of our important files we use cloud services provided by Dropbox, Google Drive. But to store and back up

our codes we use GitHub. Git organizes our files into a repository. A repository can contain any number of files organized the way we want. A single repository usually contains all the code used by a single project. Git stores a copy of the repository on a local machine. But to submit and to backup your work, you are going to push your changes to the GitHub, Git hosting service.

D. JENKINS

Jenkins is both a Continuous Integration and Continuous Deployment tool [6]. Jenkins supports a variety of configuration management tools such as Ansible and Chef. We have already introduced this Configuration management tool as individual components. However, Jenkins can be used to call Chef cookbooks or Ansible playbooks to facilitate the automated delivery of infrastructure configuration [17]. Jenkins is used more often in software development for building deployments, it supports many plugins that enable anything from deploying scripts to launch virtual machines through VMWare or Vagrant to Docker containers across environments.

Jenkins allows to create and 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 [6]. The variety of plugins you can download and install as well as the way you can manage and distribute the building load across multiple slave servers ensure that you have a robust and performant environment for automating all facets of your build, integration and deployment process.

E. ANSIBLE

Ansible is a configuration management tool like other tools such as Chef, Puppet and Salt [2] [4]. Ansible is agent-less which means that no additional software is required to be installed on the target machines. Ansible is written in Python and uses Playbooks written in YAML. In addition, it relies on SSH to connect to the managed hosts and on host configuration file in which the hosts to be managed are placed. In addition, password-less login via SSH can be created which allows. To install Ansible on any machine you need to first install Python because python is the prerequisite to install Ansible as it need python libraries to run. Ansible to run in an unattended mode of sorts. Ansible works in two modes: ad-hoc line and playbook files. In ad-hoc line mode, you pass the instructions to the remote servers as arguments and parameters to an ansible command. In the playbook files mode, you type those instructions in a YAML file. YAML is an abbreviation for Yet Another Markup Language. It is used the same as JSON files to serialize data in an easy format that is readable by both humans and machines.

Ansible uses */etc/ansible/hosts* file to describe the machines you want to be managed or configured by Ansible [1]. This file contains machine hostnames, IP addresses or a mix of them. This depends on how the server machine (the one with Ansible installed) can communicate with target hosts. Hosts can be added either in named groups for easier reference or as stand-alone entries. The Ansible hosts files support in a variety of keywords. The most common use is a simple file contains hostnames using password-less SSH.

IV. CI/CD PIPELINE

CI/CD helps increase the velocity of development. This helps to finish the development process faster but will also make adding new features and resolving software defects much faster.

Using Continuous Integration (CI), you pull together the various software components-build and test them as a single unit. Defects must be addressed immediately and are the reason for real-time feedback in builds. The artifacts created by a CI/CD process must always result in a clean build. In CI, the application should not stay overnight in a non-stable state. This means that any software defects which occurs after the commit must be resolved as soon as possible. So, the time to fix bugs will be kept to a minimum as every minor failure will be solved early enough before causing more severe bugs. Developers will always be able to pull a clean copy of the application from the repository; as any failing code will not be allowed to be pushed to the central repository. No human intervention exists in the operation, so there are very fewer chances of manual error which increases the efficiency. Special tools are designed for this such as Jenkins. Combined with a test framework like PHP Unit.

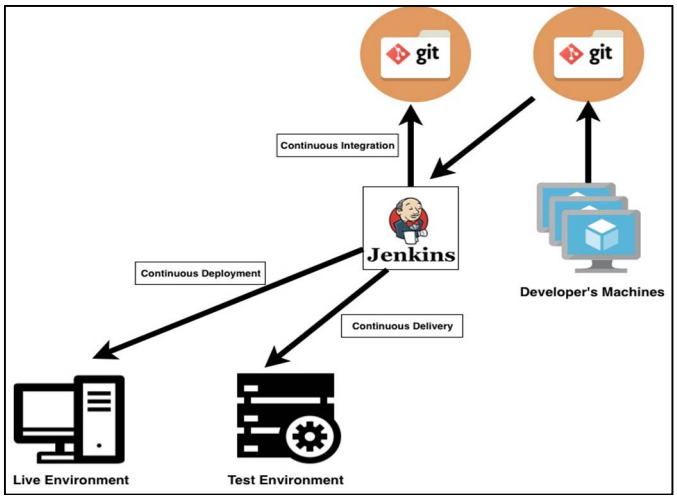

Fig. 3: CI/CD Pipeline

- Continuous Integration: It is the one with least concern as it has the minimum automation [16]. CI stops when the committed code is tested and integrated successfully into the master branch. CI has very less automation
- Continuous Delivery: Automation is a more than CI and delivers the application code from the master branch to the non-production servers instead of having to pull the changes manually from the repository
- Continuous Deployment: Over here, full automation is achieved. Code is deployed by the application code or to the binary artifacts and is given to the live production servers to be consumed by end users

V. CREATING DEVOPS INFRASTRUCTURE

We are building an environment to demonstrate the power of different DevOps tools. Git for the version-control system and source-code management. Vagrant to build and configure the lightweight development infrastructure [18]. Docker to create and run distributed applications or microservices inside the isolated container. Ansible for configuration management, application deployment and task automation. Jenkins [17] to centralize all the testing environment by building CI/CD pipeline which includes triggering auto builds, auto-promote builds from one environment to another, code analysis, auto version, etc. In this DevOps infrastructure, we are using Vagrant to spin multiple Virtual Machine's (5VMs) on Virtual Box [10]. To further configure the VMs we have Ansible-playbooks to install plugins and packages. A client machine is where all the necessary codes is stored and manually pushed to the GitHub repository [13]. The Web Server is hosting and running the application. A Database Server is acting as a data repository for the application. To test any update in the code, we are using a CI/CD pipeline inside a Jenkins Server. We have created 3 branches: Feature, Integration and Master branch on the GitHub [1]. Upon successful testing, the code is pushed to different branches for rigorous testing and after finishing all the different tests, the code is pushed back to the Web Development Server for the last and final test before it is pushed to live Web Server [10]. Once the development and operation team are satisfied with the updated version of the application, it is finally live on the original Web Server. In case of any sudden failure, it will only take a few minutes to make a new server up and running. Also, all these tools give us the leverage to achieve consistency in the infrastructure. This is very vital in a big network to avoid any version conflicts.

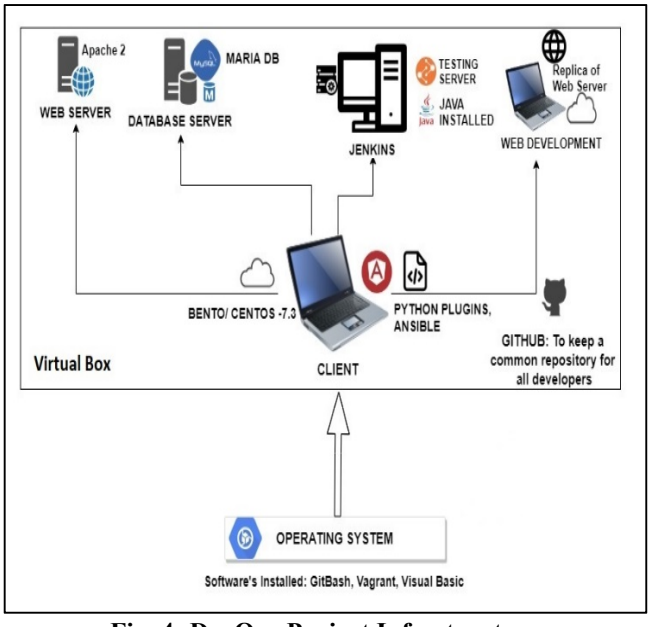

Fig. 4: DevOps Project Infrastructure

VI. ADVANTAGE

By implementing a culture of DevOps, it initiates and promotes repeatability, measurement and consistency. Implementing automation naturally improves the velocity of change, increases the number of deployments a team can do in any given day and improves time to market [2]. A byproduct of automation is that the mean time to resolve will also become quicker for infrastructure issues. If infrastructure or network changes are automated, they can be applied much more efficiently than if they were carried out manually [11]. Manual changes depend on the velocity of the engineer implementing the change rather than an automated script that can be measured more accurately. Utilizing automation and effective monitoring also means that all members of a team have access to see how processes work and how fixes and new features are pushed out.

VII. CONCLUSION

In a nutshell, DevOps brings a holistic approach to the complete business delivery system. DevOps combines developers and operations team to integrate people, processes and technology for an automated software delivery that is agile, scalable and cost-effective.

References

[1] DevOps for Networking by Steve Armstrong, PackT Publications, first edition, 2016

[2] DevOps; Puppet, Docker and Kubernetes – Learning path by Thomas Uphill, Arundel, Khare, Saito, Lee and Carol Hsu, Packt Publications, First Edidition, 2017

[3] DevOps Troubleshooting – Linux Server Best Practices by Kyle Rankin, 2013

[4] Bringing DevOps for Networking with Ansible by RedHat Enterprise

[5] Documentation of 'PaaS and Container clouds' by John Rofrano, IBM, NY Univeristy, 2015

[6] Website: https://jenkins.io/doc/

[7] Website: https://git-scm.com/book/en/v2

[8] Website:https://www.atlassian.com/git/tutorials/what-is-version-control

[9] Website: https://www.learnenough.com/git-tutorial

[10] Projects in Devops: Build real world Processes by Edunoix Learning Solutions, Online Web Courses

[11] Slides of 'DevOps for NetOps' by Rick Sherman from PuppetLabs

[12] Documentation of 'Devops defined' by HashiCorp.Inc

[13] Webiste: https://nvie.com/posts/a-successful-git-branching-model/

[14] Lecture Slides of DevOps for Networking by Prof. Widhlam, SMU,2018

[15] Docuemtation of 'Understanding in DevOps' by RedHat Enterprises

[16] Cloud Native Applications- The Intersection of Agile Development and Cloud Platforms by Douglas Bourgeois, David Kelly, Thomas Henry mebers of Delloitte Touche Tohmatsu Limitted,2016

[17] Article on 'How to Use the Jenkins Scripted Pipeline' by Alejandro Berardinelli, Blazemeter

[18] 'Vagrant Up and running' by Mitchell Hasimoto, O'reilly Pubblications, October Edition, 2013

[19] 'Quick start to Vagrant on Windows 10' by Onur Baskrit, SW Test Acamdey,2017

[20] 'How to create a Docker repository in Artifactory' by Laszlo Pinter of Pinter Computing, 2017

[21] 'See What's New in GitKraken' by Sara Breeding, DevOps Zone, Aug'2018