Brife Critical Analysis of DevOps Tools

Prof.Shantala .C.P[1] Akshaya H L[2] Veena K[3]
[1]Vice-Principal & HOD [2,3]Student
[1,2,3]Department of Computer Science and Engineering
[1,2,3]VTU.CIT. Gubbi, Tumkur

Abstract— DevOps is a practice in which operations and developers work together in each stage of DevOps life cycle (from development stage to production stage).Devops is good for developers. Developers need DevOps because for a better quality of life, pride of ownership and for more relevant work. Devops can be divided into four categories like log monitoring, monitoring, build and test and deployment and configuration. In this paper, authors give the brief introduction to DevOps Build and Test tools, Deployment and configuration and comparisons of tools.

Key words: Log Monitoring; System and Network Monitoring, Bulid and Test, Deployment and Configuration

I. INTRODUCTION

Devops is a new trend which has emerged from collision between two old trends namely 'agile system administrations' or 'agile operations' and other is the understanding the value of collaboration between development and operation staff in each and every stages of DevOps life cycle. 'Dev' means all the developers involved in production stage. 'Ops' means the term used for system engineers, system administrators, security professionals and various other disciplines or job title. So, totally 'DevOps' does not differentiate between any of the system administrator sub-disciplines. DevOps is a practice in which operations and developers participate together in each stage of DevOps life cycle like from development stage to production stage.

The utilization of DevOps strategy and an organized procedure for incorporating security into the advancement procedure is turning out to be more pervasive as huge ventures are seeing the advantages of a vital union between improvement groups and operations. As opposed to tossing the pig over the wall and trusting it transforms into bacon when it touches the ground in operations, the relationship between the two warring groups is evolving.

A. Log Monitoring:

Logs are used to analyze the system performance and usage trends. So log data are used by developers in debugging process. When any application comes across different stages of DevOps life cycle (like testing, business analytics, production monitoring), log acts as like a valuable tool [4]. When log data applied to first stage of DevOps life cycle, log acts like debugging tool and also serves as system load and performance testing [4]. When log data applied to second stage of DevOps life cycle, logs are used for production monitoring and production trouble is shooting [4]. When log data applied to third stage of DevOps life cycle, logs are used for web analytics and business metrics [4]. When we analyze the logs it improves the QA process by catching the issues faster, identifying the issue before they grow and also having better communication between the groups [5]. Logs are considered to be critical while measuring the success of the service and also trouble shooting the issues when aroused [6]. Real scalability and security are the features that don't receive attention while managing the logs. There number of Devops log tools for managing the log data. In these some of them are open source and some of them are paid tools.

B. Monitoring:

Monitoring tools will be used by the developers to make sure that the deploying software is performing correctly [1]. DevOps monitoring tools can be classified as system and network monitoring tools. System monitoring tools will be used for monitoring the performance of the system, collecting and storing the data, producing graphs of the collected data and it keeps track of system resources [2]. Network monitoring tools can be used to monitor the network and its resources, checks for the status of the network devices and if in case any issue arises it will notify the administrator, it can also collect and analyze the network traffic [3].

C. Build and Test:

The application of these tools is in automating common developer tasks like compilation of source code into binary code, creation of executable, running test cases and in the creation of documents. Tools falling under this category are Ant, Lvy, Gradle, Jenkins, cruise control and Maven. Hosted services like Travis CI offer additional options. The correct tool chain for DevOps will automate IT services, provide real-time visibility into system and application performance and also gives us a single source of truth. More important than an individual tool's capabilities, still, is how closely it all matches the organization's strategic goals. That's the method to maximize the chances of attaining DevOps goodness [7].

D. Deployment and Configuration:

System administration main task is of helping people to use computers. System administrators are domain experts who provide impedance matching between users' desires and computers. The expertise of system administrators is manifested in their choices of computer hardware and software and of system configuration. Environment of work places are constantly changing; the need for timely software updates and frequent configuration changes. Configuration and Deployment tools provide different levels of automation out of all have the same basic goal: to help in the system configuration & deployment process.

II. DEVOPS TOOLS

A. Build and Test:

1) Jenkins:

Jenkins is one open source tool to perform continuous integration. It was split from Hudson after a dispute with Oracle. The basic functionality of Jenkins is to perform a predefined list of steps based on a certain trigger. Jenkins is

an award-winning application that monitors executions of repeated jobs, such as building a software project [8].

Jenkins also monitors the execution of the steps and allows stopping the process if one of the steps fails. Jenkins allows notifying users about the build success or failure.

Jenkins is simple and easy to install, understand and use, the truth that it is Java-based must not be an impediment to .NET development shops. Jenkins provides an impressive browser-hosted project management dashboard.

Jenkins can be started via the command line or can run in a web application server. Under Linux you can also install Jenkins as a system service. It supports file fingerprinting, it also support always security performance [8][9]

2) Gradle:

Gradle is a project automation tool that builds upon the concepts of Apache Ant and Apache Maven and is licensed under the ASL. Introduces a Groovy-based domain-specific language (DSL) instead of the more traditional XML form of declaring the project configuration. Unlike Apache Maven, which defines lifecycles, and Apache Ant, where targets are invoked based upon a depends-on partial ordering; it uses a directed acyclic graph to resolve the order in which tasks can be run [7].

Gradle build scripts are written in Groovy, not XML. But different other approach this is not for simply exposing the raw scripting power of a dynamic language, It Architecture contains Deep API, Plugins, Build tools and Libraries. It is the first build integration tool, it supports Easy of migration, Groovy, wrapper, scales and Multiproject builds. Also including CD optimization, Performance tuning, Standardization, Plug-in development.

3) Maven:

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model, which can manage a project build, reporting and certification from a central part of information.

Maven's main goal is to allow a developer to know the complete state of a development effort in the shortest period of time [7].

Maven does encourage best practices, but we realize that some projects may not fit with these ideals for past reasons. Even as it is designed to be flexible, to an extent, in these situations and to the needs of different projects, it cannot provide to every situation without making compromise to the integrity of its objectives.

It contains Core Engine provides project processing, Build life cycle management, Framework for plug-ins, Plug-in provides the core operation to build user projects and plug-ins also provides one or more goals and Repositories[7].

4) Ant and lvy:

Apache Ant is a Java library and command-line tool whose work is to drive processes described in build files as targets and extension points dependent upon each other. The major known usage of Ant is the build of Java applications. Ant supply a number of built-in tasks allowing to compile, assemble, test and run Java applications [7].

Ant is written in Java and Ivy is a very powerful dependency manager oriented toward Java dependency management, although it can be used to manage dependencies of any kind. Software development projects look for a result combine build tool and dependency management can use Ant in grouping with Ivy.

It contains number of main features like Clean dependency reports, Non-intrusive, Extremely flexible, Easily extensible, Transitive dependencies, Strong conflict management, Out of the box maven repository support.

5) Cruise control:

CruiseControl is an open source tool setup specifically to perform continuous integration software builds.CruiseControl architecture helps to easily understand the modules functionality [10].

CruiseControlis composed of 3 main modules: The build loop: center of the system, it triggers build cycles at that time notifies different listeners using a variety of publishing techniques. The trigger can be internal or external. It is configured in an xml file which maps the build cycles to certain tasks, thanks to a system of plug-in. Depending on configuration, it can produce build artifacts. The JSP reporting application allows the user to browse the results of the builds and access the artifacts. The dashboard provides a visual representation of all project build statuses

B. Deployment and Configuration Tools

1) Puppet:

Puppet tool is a configuration management system that allows you to define the state of your IT infrastructure, then without human intervention enforces the correct state. Whether to control just a few servers or thousands of physical and virtual machines, this tool automates tasks that sysadmins often do manually, freeing up time and mental space so sysadmins can work on the projects that deliver greater business value. Whether you're deploying vendor-supplied applications or working with a team of internal software developers, Puppet automates every step of the software delivery process: from provisioning of physical and virtual machines to orchestration and reporting; from early-stage code development through testing, production release and updates. Puppet ensures constancy, reliability and stability. It also facilitates closer association between sysadmins and developers, enabling more efficient delivery of cleaner, better-designed code [16].Once you install Puppet, every node in your infrastructure has a Puppet agent installed on it. Also have a server designated as the Puppet master.

2) Chef:

Chef is a configuration management and automation platform from Opscode. Chef helps to describe the infrastructure with code. Since infrastructure is managed with code, it can be automated, tested and reproduced with ease.

Chef is a powerful automation platform that transforms complex infrastructure into code, bringing servers and services to life. Whether the user operating in the cloud, on-premises, or a hybrid, Chef automates how applications are configured, deployed, and managed across your network, no matter its size.

Chef is a thin DSL (domain-specific language) built on pinnacle of Ruby. This approach allows Chef to provide just enough abstraction to make reasoning about your infrastructure easy. Chef includes a built-in taxonomy

of all the basic resources one might configure on a system, plus a defined mechanism to extend that taxonomy using the full power of the Ruby language. Ruby was chosen because it provides the flexibility to use both the simple built-in classification, as well being able to handle any customization path that organization requires[17].

3) Rancid:

RANCID monitors a router's or device configuration, including software and hardware (cards, serial numbers, etc) and uses CVS Subversion to maintain history of changes.

RANCID also includes looking glass software. Our version has added functions, supports cisco, juniper, and foundry and uses the login scripts that come with rancid; so it can use telnet or ssh to connect to your devices[18].

4) CFEngine:

CFEngine is a suite of programs for included autonomic management of either individual or networked computers. It has existed as as software suite since 1993 and this version published under the GNU Public License (GPL v3) and a Commercial Open Source License (COSL). CFEngine is Copyright by CFEngine AS, a company founded by CFEngine author Mark Burgess.

CFEngine is an open source configuration management system, written by Mark Burgess. Its primary function is to provide automated configuration and maintenance of large-scale computer systems, including the unified management of servers, desktops, consumer and industrial devices, embedded networked devices, mobile smartphones, and tablet computers [19].

5) Ansible:

Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.

Ansible's goals are foremost those of simplicity and maximum ease of use. It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport (with an accelerated socket mode and pull modes as alternatives), and a language that is designed around auditability by humans – even those not well known with the program.

Ansible is appropriate for managing small setups with a handful of instances as well as enterprise environments with many thousands.

Ansible manages machines in an agentless manner. Ansible is decentralized – it relies on your existing OS credentials to control access to remote machines; if needed it can easily connect with Kerberos, LDAP, and other centralized authentication management systems[20].

III. COMPARISON OF TOOLS

A. Build and Test Tool

1) Build Tool

- Maven, Gradle, Ant and Lvy are Free and open source tools.
- SCM support for Maven, Gradle, AntantLvy.
- Build Life cycle only in Maven and Gradle whereas this is not in rest of the tools.
- Gradle tool use ivy dependency management (which uses maven repositories). Dependency downloads are quite fast.
- Only maven, Gradle support Jetty plugins and jetty server.
- Gradle tool Integrates ant tasks also.
- Gradle tool only supports the standard directory layout (So for compilation, testing and packaging no extra tasks need to be written as long the files are in the layout).
- Gradle tool Building is fast compared to ant, lvy and maven.
- Only Maven, Gradle and Lvy support plugins reusable features.
- Gradle Third party plugin support is not as good as maven.
- Only gradle tool support standard Junit reports.
- IDE support for Maven, gradle, ant and lvy.
- Gradle tool only based on groovy, scripting tasks are quite because it is in a programming language.
- Lvy only provide clean dependency report.
- Gradle Build scripts are code format.
- Gradle tool only support Dynamic task creation and modification tasks.
- Gradle tool only support Powerful logging and logging configuration.
- Gradle tool Speed is very high.
- Maven and gradle only have better dependency management.
- A CI server supports all tools.

2) Test Tool

- Jenkins, Cruise Control, Hudson, Travis CI are Free and open source tools.
- SCM support for Jenkins, Cruise Control, Hudson, Travis CI.
- Jenkins can onlysupport distributes build/test loads to multiple computers. This lets you get the most out of those idle workstations sitting beneath developers' desks
- Jenkins only gives you clean readable URLs for most of its pages, including some permalinks like "latest build"/"latest successful build", so that they can be easily linked from elsewhere.
- Jenkins, Hudson, Travis CI tool supports Emma output analysis.
- Jenkins, Hudson, Travis CI tool supports FindBugs output analysis.
- All tools support Check style output analysis.
- Jenkins can only generate a list of changes made into the build from Subversion/CVS. This is also done in a fairly efficient fashion, to reduce the load on the repository.
- All tools support RSS/E-mail/IM Integration.
- Jenkins, Hudson, Travis CI tool supports Disk usage analysis.
- Jenkins, Hudson, Cruise Control tool supports Bugzilla integration.
- Only Jenkins, Hudson, Travis CI tool supports JUnit output analysis.
- All tools support Plugin Support.
- File fingerprinting option only in Jenkins tool.
- Only Jenkins, Hudson, Travis CI Build history management.
- All tools support Ant, JDK and Maven.

- Jenkins use servlet container such as Apache Tomcat, Cruise Control use Container technology like Jetty, Hudson Comes with Winstone and Travis CI use Routing your build to container-based infrastructure.

3) Configuration and Deployment Tools:

- All the tools under the configuration and deployment are Open Source Tools, versioning support ,LINUX,Solaris & Windows mainly, high scalability, Security.
- Salt stack and Rancid has the same approach of Infrastructure as code, where as other tools have different approaches like Puppet-Declarative & Model Based, Chef-Code based, CFEngine-language based, Ansible-state driven resource model.
- Language used in the Puppet tool is Ruby, Chef-Ruby& Erlang , CFEngine-C, Ansible and Saltstack-Python.
- With respect to the community of usage, Puppet, CFEngine, Ansible, Salt Stack-larger user base, rancid & chef-comparatively small user base.
- Puppet has no extended API, but other tools have an extended API.
- Code Execution for chef & Ansible is at client side and all other tools is Client & server.
- Installation procedure for the Puppet is Automation, and all other tools is manual.
- Interface supported by the Puppet is Graphical & command line, CFEngine &Rancid-command line, Ansible & SaltStack is graphical.
- Time & Resource consumption by Puppet is more compare to CFEngine.
- Workflow In Puppet, CFEngine and Ansible is by coordination of distributed changes.
- Ansible is easy to use, whereas CFEngine& Puppet – Medium, Chef – Hard.
- Puppet provides Translation agent as centralized management, but CFEngine provides strongly distributed management.
- Type of grouping used by Chef is static, query based & Puppet, CFEngine uses Static, Query based, Hierarchical.
- All the tools provide system device support excluding Rancid.
- Puppet, CFEngine,Rancid provides limited network device support.
- Chef uses the language which is DSL based and rest are Non-DSL.

IV. CONCLUSION

As per the present survey of devops tools with respect to the various metrics, In build and test tool Gradle and Jenkins is said to be best tools. In Deployment and Configuration tool puppet is considered to be the best tool.

REFERENCES

[1] http://blogs.vmware.com/cloudops/tag/devops
[2] http://en.wikipedia.org/wiki/System_monitoring
[3] http://en.wikipedia.org/wiki/Network_monitoring
[4] http://devops.com/features/log-data-valuable-tool-devops-lifecycle-beyond/
[5] https://blog.logentries.com/2014/12/connected-qa-selenium-log-analysis/
[6] https://www.loggly.com/blog/loggly-qa-talking-james-urquhart-continuous-integration-deployment-devops-role-log-monitoring/
[7] http://zeroturnaround.com/rebellabs/java-build-tools-part-2-a-decision-makers-comparison-of-maven-gradle-and-ant-ivy/
[8] https://wiki.jenkins-ci.org/display/JENKINS/Jenkins+Best+Practices
[9] http://en.wikipedia.org/wiki/Jenkins_%28software%29
[10] http://cruisecontrol.sourceforge.net/overview.html
[11] Tim Heckel (18 Feb 2013). "Meet Travis CI: Open Source Continuous Integration". *InfoQ*. Retrieved 28 June 2013.
[12] http://java.dzone.com/articles/continuous-integration-aint
[13] http://www.drdobbs.com/jvm/why-build-your-java-projects-with-gradle/240168608
[14] https://docs.oracle.com/middleware/1212/core/MAVEN/maven_version.htm#MAVEN8855
[15] https://blog.newrelic.com/2014/06/02/devops-tools/
[16] https://puppetlabs.com/puppet/what-is-puppet
[17] https://docs.chef.io/chef_overview.html
[18] http://www.shrubbery.net/rancid/
[19] https://auth.cfengine.com/archive/manuals/cf3-Reference
[20] http://docs.ansible.com/

