ASIAN JOURNAL OF PHARMACEUTICAL AND CLINICAL RESEARCH

INNOVARE ACADEMIC SCIENCES
Knowledge to Innovation

Special Issue (April)

Online - 2455-3891
Print - 0974-2441

Full Proceeding Paper

A REVIEW OF EXISTING CLOUD AUTOMATION TOOLS

PRASSANNA J*, ANJALI R PAWAR, NEELANARAYANAN V

School of Computing Science and Engineering, VIT University, Chennai, Tamil Nadu, India. Email: prassanna.j@vit.ac.in

Received: 23 January 2017, Revised and Accepted: 03 March 2017

ABSTRACT

Many enterprises are running distributed applications on their on-premise servers. However, if load on those servers changes unexpectedly, then it becomes tedious to scale the resources and requires skilled human power to manage such situations. It may increase the capital expenditure. Hence, many companies have started to migrate their on-premise applications to the cloud. This migration of the applications to the cloud is one of the major challenges. To setup and manage the growing complex infrastructure, after migrating these applications to the cloud are really a time-consuming and tedious process which results in downtime. Hence, we need to automate this environment. To achieve architecture for the distributed systems which support security, repeatability, reliability, and scalability, we require some cloud automation tools. This paper summarizes tools such as Terraform and cloud formation for infrastructure automation and Docker and Habitat for application automation.

Keywords: Cloud computing, Infrastructure automation, Application automation.

 DOI: http://dx.doi.org/10.22159/ajpcr.2017.v10s1.20519

INTRODUCTION

Compute, storage, and network are the basic computing resources. The use of virtualization has reduced the time required to deploy computing resources from weeks to few minutes. However, for building a cloud-based infrastructure, this is not enough. Hence, the deployment and maintenance of those resources should be automated and managed in such a way that the resources can be effectively used, rapidly provisioned, and released with less management efforts.

To reduce the capital expenditure on the computational resources, enterprises are moving their workload to the cloud with the promise that they will get access to flexible and elastic compute resources in minimal cost. However, managing this growing infrastructure manually along with the deployed application is one of the major challenges. This leads to introducing the cloud automation concept. There are various cloud automation tools which do all these work for us ensuring that all the tasks regarding deployment and allocation of the computational resources are done efficiently.

LITERATURE SURVEY

Authors in this Juve and Deelman [1] paper say that Infrastructure as a Service clouds provide the ability to provision VMs on demand, but they do not give information for managing those resources which are provisioned. Hence, to use such clouds effectively, tools are needed to use which can help users to easily deploy applications in the cloud. The authors of this paper developed a system to create, configure, and manage the CM deployments in the cloud.

Due to variety of the operating system and applications, it becomes very difficult to deploy a large number of virtual machines in a short period. This Zhang *et al*. [2] paper proposes an automatic deployment mechanism based on cloud computing platform openstack. This system is responsible for the automatic deployment at operating system level as well as application level. They have developed an interactive dashboard for the users which helps them to deploy their systems and the applications without professional knowledge of cloud.

This Callanan *et al*. [3] paper has presented the architecture of an environment migration framework for automating the migration of existing infrastructure, creation, and configuration in the cloud. They have discussed some challenges faced while migrating the applications to the cloud typically security as main along with the legal and compliance issues.

Compute, storage, and network are the primary resources of computing. Provisioning time for deployment of these resources is remarkably minimized by virtualization technology. However, to construct a cloud-based infrastructure, still only data center virtualization is not sufficient. If we go for only this data center virtualization technique, then it may generate virtual resource sprawl. In addition, the infrastructure which is cloud-based cannot construct by only virtual infrastructure. In other hand, physical infrastructure also needs to be automated. Hence, to automate and manage cloud-based infrastructure (virtual as well as physical resources), we need software. For management and automation of cloud-based infrastructure, different modules and their integration are discussed in cloud management and automation paper [4].

CLOUD AUTOMATION TOOLS

This section summarizes the study of existing infrastructure automation tools like Terraform and cloud formation and application automation tools like Habitat and Docker.

Terraform

Terraform takes the concept of managing Infrastructure as Code. This is one of the best tools for creating, configuring, managing, and versioning the infrastructure very effectively and safely. It can be used to codify the knowledge of building and scaling a service into a configuration. It uses text files to describe the infrastructure. It supports various cloud service providers. If configuration changes then Terraform will determine that changes and create execution plans according to it [5].

The key features of Terraform are:

- Infrastructure as Code
  Terraform is used for the managing cloud infrastructure as code. The infrastructure is defined using the configuration syntax. This infrastructure can be easily shared and reused for other environment.
- Execution plans
  Terraform generates execution plan which states what it will do to reach the desired state. This execution plan describes what will happen when we call to apply. Then, it executes that plan to build that infrastructure.

- Resource graph
  It parallelizes the creation and modification of any nondependent resources and builds a graph of all our resources. This helps Terraform to build infrastructure efficiently.
- Change automation
  If complex changes are required then that can be applied easily with the less human involvement. With the help of execution plan and resource graph, we will easily get knowledge of what Terraform will change and in what order by avoiding the possible human errors.

Fig. 1 shows the workflow in Terraform. Developers can use Terraform plan command to calculate what changes will be performed after using Terraform build command. Hence, they can update the configuration according to the plan. If Terraform build command is successful, then it will deploy the changes in Amazon Web Service (AWS) cloud. If you want to have backup of tfstate, then you can update the tfstate in S3. This will help other team members to get the tfstate file and make changes.

Cloud formation

Cloud formation is a service provided by AWS which helps us to setup our AWS resources. This helps us to spend more time on focusing on our application which is running in AWS rather than spending time on managing those applications. We can create a template file which states the resources we want, and AWS cloud formation does the provisioning and configuring those resources for us. It handles the creation and configuration of AWS resources as well as it figures out what is dependent on what [6].

We can create and provision AWS infrastructure deployments predictably and repeatedly with the help AWS cloud formation. It enables us to use AWS products such as Amazon EC2, Amazon SNS, Amazon elastic block storage, auto scaling, and elastic load balancing to build highly scalable and reliable applications in the cloud by providing less attention to the underlying infrastructure [6].

The key features of AWS cloud formation are [7]

- Easy infrastructure management
  To build highly scalable and reliable application, we might require an auto scaling group, an elastic load balancer, or any other service from AWS. Hence, after we create the resources, we have to configure this to work together. If we are doing this manually, then it adds the complexity and time before we get our application up and running. Instead of this, we can modify an existing AWS cloud formation template which describes all our resources. It provisions all the required resources for us and easily manages a collection of resources as a single unit.
- Easily reuse your infrastructure
  To make application more reliable, we might replicate it in multiple regions so it becomes available all the time. AWS cloud formation enables us to reuse the template to set up the required resources consistently and repeatedly in multiple regions.

Easily track and control changes to infrastructure: We are using templates to describe the infrastructure resources. These are text files

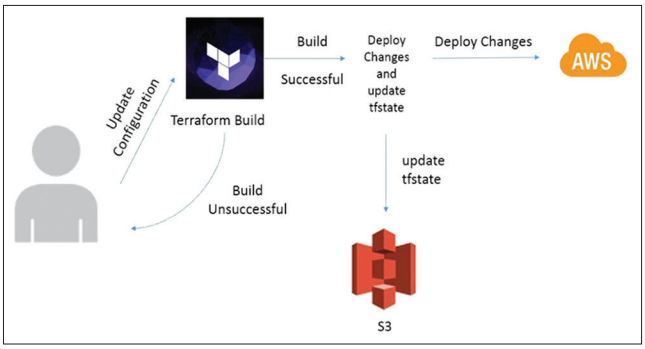

Fig. 1: Sample workflow in Terraform

so we can easily track and control the changes to our infrastructure. We can use a version control system along with the templates so we will get to km = now what changes we made and if at any point I want to rollback our infrastructure to the original settings, we can use previous version of our template.

Docker

Docker is an open-source tool which is designed for creating, deploying and running applications easily using containers. Docker containers allow developers to wrap an application with all required libraries and other dependencies and make it as a single package. This gives the guarantee that the software will always run the same without caring of its environment. Docker gives flexibility to developers to build, ship as well as run any application, anywhere [8].

Docker is bit similar to the virtual machine. However, instead of creating whole virtual operating system, Docker enables applications to use the same underlying kernel. This improves the performance boost significantly and reduces the size of the application. It is designed to benefit both developers as well as system administrators. So that, developer can focuses on writing code without underlying infrastructure. As Docker is open-source, anyone can contribute and extend it to fulfill their own needs. Hence, it helps developers to start using one of thousands of programs which are already designed to run in a Docker container. Because of its low operation expense and small footprint, operations staff gets flexibility to reduce the number of system needed [10,11].

The key features of Docker are:

- Agility
  Docker can build any application in any language using any stack and dockerized applications can be run anywhere. Hence, this gives flexibility to developers to define environment and ability to create applications.
- Portability
  Docker allows enterprises to make the best business decision by choosing any infrastructure such as cloud, virtual machines, or bare metal servers. Docker can run applications anywhere on any infrastructure.
- Lightweight
  All the containers are running on same machine share the same operating system kernel. Images are built from the layered file system hence shares the common files and make efficient use of disk usage.
- Secure by default
  Containers provide an additional layer of protection for the application by isolating application from one another and the underlying infrastructure.

Docker chooses client-server architecture model. The Docker client is the basic interface to the Docker. It will take configuration and commands flags from the user and interacts with the Docker daemon. Single client has capability to communicate with multiple unrelated daemons as well as the remote Docker daemon. This Docker daemon runs on the host machine. Docker daemon does the work of building, running, and distributing your Docker containers.

To run multicontainer applications, Docker has provided tool like Docker-compose. With Docker-compose, we can write compose file which describes a configuration for application services. Using compose, we can define multiple isolated environments on a single host. It can be used in various ways such as in development environments, automation testing environment, or in single host deployment [9].

The use of compose is a three-step process [9]:

1. We have to define our environment using Dockerfile so that it can be used anywhere.
2. Use Docker-compose. YML file to define the services which make up our application so that they will run together in and isolated environment.

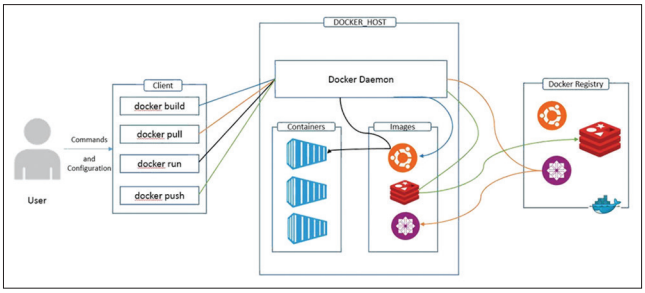

Fig. 2: Simple Docker architecture [9]

3. Finally, use Docker-compose up command and compose will run our application.

If you want to use compose in production environment, then we have to make slight changes in configuration and we can rebuild the image and recreate the application containers. This will help use to deploy application on single server. If you want to your applications on multiple hosts, you can use compose against swarm instances [12]. Docker Swarm creates a pool of the Docker hosts and turns it to single virtual host. Hence, any tool which has already interacted with Docker daemon can use Swarm to scale to multiple hosts. It supports tools such as Docker-compose, Docker Machine, and Dokku and Jenkins [13].

Fig. 2 shows how the commands are working and what changes they will perform. When user asks for Docker build then, it will build the image. When Docker pull command is given, Docker daemon will pull the image from Docker registry to the Docker host. Docker run command will use the image and creates a container from it. Docker push command is used to push the image from host to Docker hub.

Habitat

Habitat is an open-source project which provides new approach to automation. It focuses on application rather than infrastructure, it runs on. Using Habitat, the application we build, deploy, and manage will behave consistently in any runtime. It packages our application and its automation together. It enables us to ship our application as well as the automation we need to manage to any platform [14].

Habitat helps us to spend less time on environment and more time on building features. It puts application first. It packages application code, runtime dependencies, start-up scripts, and configuration together [15].

The key features of Habitat are [16]:

- Run any application, anywhere
  With the help of Habitat application can run in any environment whether it is container, bare metal, or PAAS.
- Easily port legacy applications
  The legacy applications become independent of the environment, for which they were designed when they are packaged in a Habitat. They can easily adopt modern environments such as cloud and containers.
- Improve the container experience
  The complexity of managing containers in production environment is reduced using Habitat. Habitat solves the challenges developers face when moving container-based application from development environments into production as it automates the application configuration within a container.

CONCLUSIONS AND FUTURE WORK

This study of cloud automation tools defines the importance of automation tools to achieve architecture for the distributed systems. The future work includes the deploying and managing the infrastructure and applications; on top of that using, these cloud automation tools analyze the security, repeatability, reliability, and scalability impacts on the deployed distributed system.

REFERENCES

1. Juve G, Deelman E. Automating Application Deployment in Infrastructure Clouds. Cloud Computing Technology and Science (CloudCom). IEEE Third International Conference on. Athens: IEEE; 2011. p. 658-65.
2. Zhang R, Shang Y, Zhang S. An Automatic Deployment Mechanism on Cloud Computing Platform. Cloud Computing Technology and Science (CloudCom). IEEE 6th International Conference on. Singapore: IEEE; 2014. p. 511-8.
3. Callanan S, O'Shea D, O'Regan E. Automated Environment Migration to the Cloud. 27th Irish Signals and Systems Conference (ISSC). Londonderry: ISSC; 2016. p. 1-6.
4. Wibowo E. Cloud Management and Automation. 2013 Joint International Conference on Rural Information and Communication Technology and Electric-Vehicle Technology (rICT and ICeV-T). Bandung: rICT and ICeV-T; 2013. p. 1-4.
5. Terraform. Available from: https://www.terraform.io/.
6. Cloud formation. Available from: https://www.aws.amazon.com/cloudformation/.
7. Cloud formation. Available from: http://www.docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html.
8. Docker. Available from: https://www.opensource.com/resources/what-docker.
9. Docker compose. Available from: https://www.docs.docker.com/compose/overview/.
10. Docker. Available from: https://www.docker.com/.
11. Available from: https://www.devops.com/2014/11/24/docker-vs-vms/.
12. Docker compose in production. Available from: https://www.docs.docker.com/compose/production/.
13. Docker Swarm. Available from: https://www.docs.docker.com/swarm/overview/.
14. Habitat. Available from: https://www.habitat.sh/.
15. Habitat. Available from: https://www.blog.chef.io/2016/06/14/introducing-habitat/.
16. Available from: https://www.blog.chef.io/2016/06/14/chef-launches-habitat-new-open-source-project-to-automate-applications.