Demo

IMPROVING DEPLOYMENT PROCESS

Xport Program

Abbreviations and Acronyms	
CM	Configuration Management
IAC	Infrastructure as Code
EC2	Elastic Cloud Computing
RDS	Relation Database Service
AWS	Amazon Web Services
HCL	HashiCorp Configuration Language
VCS	Version Control System

CONTENTS

1 INTRODUCTION

1.1 PURPOSE

The purpose of this document is to describe the technical details of the demo presented as the conclusion of the Xport Program.

1.2 SUMMARY

In this document I present an improvement for the process we used to work with in a past project where the configuration of all the servers was done manually and the EC2 servers were used as common servers and the fully functionality of AWS and automating was not being leveraged as could be.

In this demo, I present a solution for automating the boring stuff using tools such as Terraform and Ansible, as well as DataDog for monitoring the instances.

2 OVERVIEW

2.1 OLD PROCESS DESCRIPTION

2.1.1 Architecture description

The infrastructure consisted of 3 EC2 instances, each one with a different purpose:

1. 1 Instance for running the application.
2. 1 Instance working as a reverse proxy redirecting the traffic from the users to the application instance
3. 1 Instance with a Database used in the application

2.1.2 Process description

Each EC2 instance was a server with a Linux distribution installed that was configured manually and there was no an efficient way to set up a new instance in case of any failure with any of them nor a standardized process to follow, so it was really complicated to escalate the architecture in case of need it and any change would cause some downtime in the service offered to the users.

Also, another important thing is that the database updates should be taken care manually by any member and there was a big risk of failure in any misconfiguration.

2.2 NEW PROCESS DESCRIPTION

2.2.1 Architecture description

The new architecture consists of an undefined number of EC2 instances, since it is managed automatically by an Auto Scaling Group on AWS and an RDS instance for the Database.

2.2.2 Process description

With this new process, I offer a solution that is easily scalable and the process for setting un the full architecture is well controlled and managed, also it is versioned so at any time I could consult the state of the architecture definition. Also, the configuration of the servers is done automatically using Ansible, so at any time a new EC2 instance is created it can be configured easily using the Ansible playbook and the infrastructure is managed by Terraform.

3 WORK

3.1 INFRASTRUCTURE DEFINITION

3.1.1 Terraform

In this improvement, **Terraform** has been used as a solution for implementing Infrastructure as Code. This provides us the opportunity of controlling the version of the infrastructure with a Version Control System such as git, as I selected on the project. **Terraform** allows us to define the infrastructure of a complete project using HasiCorp Configuration Language, also known as HCL.

3.1.1.1 What is being built?

The created infrastructure is being created on the Cloud using AWS technologies. Through **Terraform** I am building:

- An Application Load Balancer
- A Launch Configuration
- An Auto Scaling Group using the Launch Configuration created
- An RDS instance
- Security Groups for ALB, EC2 instances and RDS instance
- Internet gateway for required instances
- A VPC

Using **Terraform** the Auto Scaling Group is being attached to the Application Load Balancer using the Target Groups that are required in the process, in this way the Application Load Balancer is using the EC2 instances for serving the content to the users.

The Launch Configuration has been set up to use an AMI with Red Hat installed, which is in the Free Tier eligible Images, so the servers are using the advantages that Red Hat provides.

Another big improvement made in this process is the use of an RDS instance for the Database, in this way the team stops worrying for the configuration and maintenance of the Database since it is managed by AWS, as well as the backups that can be configured.

3.2 CONFIGURATION MANAGEMENT

3.2.1 Ansible

In this improvement, **Ansible** has been used as a solution for implementing Configuration Management. This provides us the opportunity of controlling the version of the configuration done in the EC2 instances with a Version Control System such as git, as I selected on the project. **Ansible** allows us to automate the configuration of the servers using YAML format for defining the playbooks.

3.2.1.1 How is configuration done?

The configuration is done through Playbooks, in these playbooks I define the roles of the servers to configure and making use of these roles **Ansible** knows what to install on what servers. **Ansible** allows to worry only on the ***what*** to install and stop worrying on the ***how*** to install it.

The process that is being followed by **Ansible** in this project is:

1. Installs and configures NGINX on the EC2 instances to receive the traffic before the applications does it
2. Installs GIT on the EC2 instances
3. Clone the source code of the application
4. Start up the application

All this process is done through **Ansible** and there is no need to connect to the instances to configure anything, thanks to this the process is done automatically, which will save a lot of time, with the configurations defined using **Ansible** and the process can be replicated as many times as needed.

3.3 MONITORING

3.3.1 DataDog

In this improvement, **DataDog** is used for monitoring the instances that are created in the infrastructure. With **DataDog** I can easily monitor different types of instances such as EC2 and RDS as well as the Auto Scaling Groups and the Load Balancers.

3.3.1.1 How does it work?

DataDog is installed on the EC2 instances easily by downloading the script they offer and with it, it installs an agent on the instances and this agent sends the monitoring information to the **DataDog** server, which is a hosted server provided by the **DataDog** team. Apart from this process, I only need to create an AWS Role with some specific ReadOnly permissions and configure it on the DataDog service linking this AWS Role to my account and with this, I am granting access to DataDog to monitor different types of instances using the AWS CloudWatch service.

4 RESULTS

Thanks to the implementation of these tools I am greatly improving the process of initial deployment of the full application from the step of defining and creating the infrastructure to the process of configuring the created infrastructure and setting up the application working properly in the Cloud using the AWS services.

With this improvement I am adding High availability by adding a Load Balancer on the infrastructure and a more robust and scalable infrastructure. Also, I am adding monitoring of the instances, so I can add alerts for detecting possible failures or do choices in base of the metrics collected.

APPENDIX A. GITHUB REPOSITORY

This is the URL for the repository where the source code of the IAC, CM and Application is hosted.

https://github.com/danpe91/XportDemo

REVISION HISTORY					
Ver.	Description of Change	Author	Date	Approved	
				Name	Effective Date
1.0	Creration	Daniel Pedroza	19-July-2018		19-July-2018