Automated Generation of Virtual Machine Images with Hasicorp Packer

Graduation Project 2020-2021 Spring Term

Necati Turan, Işıl Alıcı, Yaren Mısırlı
Computer Engineering Department,
Istanbul Kultur University
Istanbul, Turkey

***Abstract*—In the age of cloud computing, many organizations and are now quitting monolith systems and turning their infrastructure to immutable infrastructure to simplify the stages develop-configure-deploy and its reliability through to concept of Infrastructure as Code. The general develop, configure and deploy stages are now is more complex than ever and needs to be maintained to be more time and cost effective. To create a configurated server and platform for our services and applications we use a virtual machine image on the platforms we develop and serve. A machine image is a virtual copy of an operating system that have a pre-baked operating system. This image must have installed software, patches, network configuration which is used to quickly create new running machines for our desired purposes. Using pre-baked images provides effectivity and consistency and it reduces development times radically than any other techniques. However machine image types and their configurations change for each platform. That means we have to adapt every change and update for each platform we will used and serve our applications. With an immutable infrastructure, we create a new server each time, rather than making updates to a running server. The most efficient way to apply and use an immutable infrastructure is about making every parts of the system as a single code unit. Therefore we use the term Infrastructure as Code, containing all the necessary parts like virtual server images, specific alterations of the server. Creating all this platforms and their configurations is a very complex for developers and time and cost consuming stage. Therefore Hashicorp Packer propose a quick and decent way to solve this by creating pre-baked machine images with only json formatting file from any platform. Packer automates the creation of machine images of any kind and making it run remotely. It incorporates modern configuration management by encouraging you to use a framework like Chef or Puppet to install and configure software within your images created with Packer.**

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION

Contemporary notions of the IT industry evolved by the changes on cloud computing. Companies, and us as developers are observing fundamental changes by the emergence of cloud computing and its own changes on software development cycles.

To make our points clear, first we must understand how many basic stages are there of cloud computing.

There are broadly known three layers of clouds; (I) Infrastructure as a Service (IaaS), (ii) Platform as a Service (PaaS), and finally (iii)Software as a Service (SaaS).

To be clear about understanding of the Among these three layers, IaaS systems (e.g., AWS,Azure,GCloud,DigitalOcean,Linode) offer fundamental infrastructure configurations and tools like VMs and virtual storages, and these have got the most recognition from the IT market. [4].

Either we focus SaaS or IaaS, we must organize and automate our works and applications on the servers, especially on the cloud based organizations.

Therefore updating, managing, creating basically configurating the whole infrastructure means a lot of hardwork and that means more money and time to consume.

To achieve effectivity on time, budget and human sources we focused on Immutable Infrastructure services to achieve managing VMIs and systems by using Hashicorp Packer to automate and clarify all these steps of IaC and we will examine all these layers and their problems in this paper.

II. RELATED WORK

A. Infrastructure Types

In this part we will examine alternatives and competitors of Hashicorp Packer and we will show a basic benchmark among others.

Matej Artac & Damian Andrew Tamburri [11], compiles different fields on Infrustructure As Code principle with numerous techniques. Infrastructure design and development is about the application development lifecycle stages that examines and configures the basic principle software infrastructure needs for that software such as the AWS cloud, Azure and the other numerous and type of VMs required in each fields.

The infrastructure pattern typically require numerous installation and settings of the scripts needed to be applied can be described in 3 parts. (i) create and set the required machines (either physical in a local system or virtual in a cloud based

system) for the application itself to be applied; (ii) deploy and make the settings of the the needed software and its own configurations for those VMs; (iii) finally apply and run the required secondary services or the third party applications for the software or services to be operated on a system.

B. Deciding Infrastructure Type

The We will discuss and deeply examine Vagrant, Ansible, Terraform, Docker, Kubernetes,Cloudify , Terraform and their basic comparisons.

As we mentioned earlier, there are 2 basic types of infrastructures; **mutable** and **immutable**.

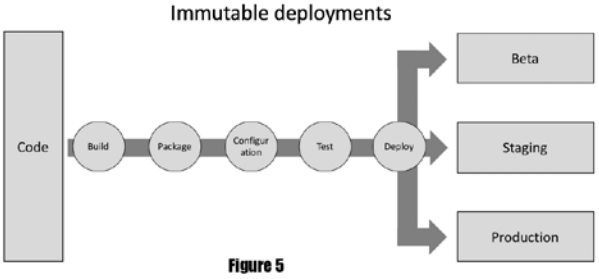


Figure 5

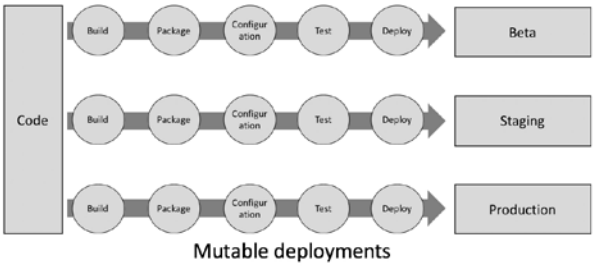


Figure 6

Mutable infrastructure uses tools such as Chef, Puppet, Ansible that automatically updates application on the server by creating a long history of changes and those are very hard to reproduce while quite difficult to examine for any errors occurring during the deployment and configuration stages.

Generally, in mutable infrastructure DevOps and staff encounter the situation where someone updated the server leading to the low quality production server and that making to downtime or unknown outcomes in the infrustructures.

Here is a figure make readers clear what all of this tools has as disadvantages and advantages.

	Chef	Puppet	Ansible	Packer	Terraform
Cloud		All	All	All	All
Type	Config Mgmt	Config Mgmt	Config Mgmt	Config Mgmt	Orchestration
Infrastructure	Mutable	Mutable	Mutable	Immutable	Immutable
Language	Procedural	Declarative	Procedural	Declarative	Declarative
Architecture	Client/Server	Client/Server	Client only	Client only	Client only
Orchestration					
Lifecycle (state) management	No	No	No	Yes	Yes
VM provisioning	Partial	Partial	Partial	Yes	Yes
Networking	Partial	Partial	Partial	Yes	Yes
Storage Management	Partial	Partial	Partial	Yes	Yes
Configuration					
Packaging	Yes	Yes	Yes	Partial [1]	Partial [1]
Templating	Yes	Yes	Yes	Partial [1]	Partial [1]
Service provisioning	Yes	Yes	Yes	Yes	Yes
[1] Using CloudInit					

Figure 7: Comparison of Tools

C. Existing Systems

As we mentioned earlier, there are 2 basic types of infrastructures; mutable and immutable.

Mutable infrastructure uses tools such as ***Chef, Puppet, Ansible*** that automatically updates application on the server by creating a long history of changes and those are very hard to reproduce while quite difficult to examine for any errors occurring during the deployment and configuration stages.

D. Mutable

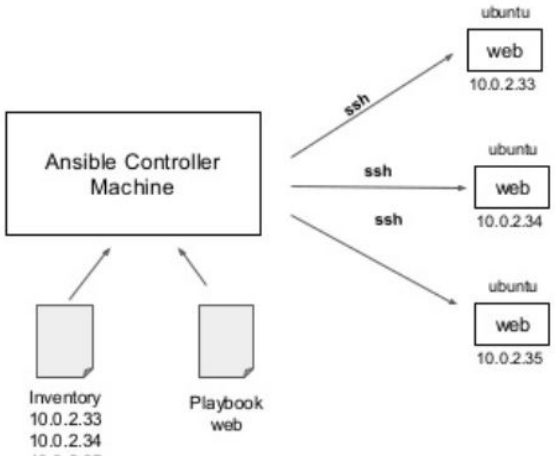
a) Ansible

Ansible is an open source tool sponsored by Red Hat, and formerly it was the simplest way to automate a server.
It's incredibly easy to use and its learning curve is quite small.
However it's not the best for the immutable infrastructures and we need other tools to use in immutable systems.
Besides its disadvantages, Ansible is quite small,fast and easy to implement.
It generates a VM by only a playbook file.(YAML)
Here is a basic installation of the Ansible on Ubuntu :

Each playbooks has plays and only have inventories and variables.
Basic components of ansible :

- Modules
- Module utilities
- Plugins
- Inventory
- Playbooks
- The Ansible search path
- Modules

In the example below, the first play aims the web servers; the second play targets the database servers.
Its syntax is easy and packages are pretty small. Relatively faster than Chef and Puppet.



b) Chef

Chef works as a configuration management tool on mutable servers. It serves a fast delivery and according to benchmark tests, it's faster than Ansible on AWS [6].

Chef is a configuration management tool and the packer is a provisioning tool.

Chef is concerned with installation and management of software on existing servers, while packer provisions the servers themselves. Therefore when using Terraform or Packer, packer is a lot better option than a configuration management tool itself.

Moreover Chef defaults to a mutable infrastructure design and making to hard to examine configuration bugs, but packer checks every change as a deployment of a new service configuration.

Chef uses an imperative language.

III. METHODOLOGY

In this section we will show manual manipulation of the servers, setting cloud based VMIs and their configurations while we also show how easy it will be with Hashicorp Packer.Packer is an extremely small and fast tool and it has a large community it as part of a CI/CD pipeline and configuration management.Within Packer's abilities to reach in cloud systems, it's easy to implement system independent construction, we choose EC2 to prove our points. Therefore we must understand the Amazon Web Services' EC2 console and its applications first.

III.I. Design Overview

Contemporary cloud services like AWS have built-in pre-baked VMs to serve customers desires. Preloading all needed software,patches,settings and configurations on an EC2 instance, then creating an image of that takes a couple of minute for each time. The resulting image can then be used to launch new instances with all software and configuration pre-loaded.This process allows the EC2 instance to come online and be ready for serve in a quick way. It doesn't only simplifies deployment of new instances but also especially handy when an instance is part of an Auto Scaling group and is it's reacting to an error in load of the system. However, if the instance creation&configuration takes longer time than it needed to be then it loses its advantage on the aim of dynamic scaling in the cloud & service. To this purpose we used Packer to pre-bake AMIs without interfering EC2 console. For this purpose; let's take quick view on AWS and EC2 console.

Steps of the implementations on AWS;

1. Set up your environment
2. Getting security Details
3. Choosing platforms and getting Pair keys as PEM file
4. Configuring IAM credentials
5. Running a Packer automation workflow

For this purpose, let's take a quick look on EC2 with figures and tables.

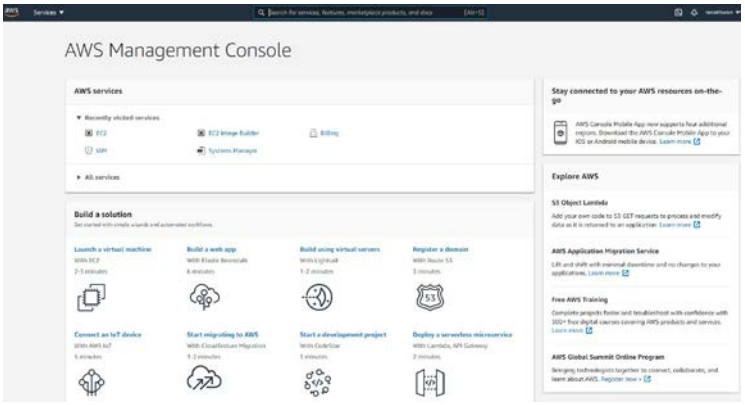


Figure 10 : AWS Management CONSOLE

III.II. MODULE A

Details On EC2

To make our points clear, we want to make readers understand why we choose Amazon Web Services.

Therefore let us line up some important features of EC2 that we will use on this paper.

- VMIs, known as instances that implies virtual computing&configurating environments on cloud
- Pre-baked VMI and configuration templates for our copies (instances), known as AMIs, that whole collections, the requirements for our need,
- Numerous configurations of CPU, memory, storage, and networking volume for our instances, known as instance types
- Secure login information for your instances using key pairs so we can secure our services
- Storage capacities for temp data (deleted when we stop/hibernate/terminate our instance)

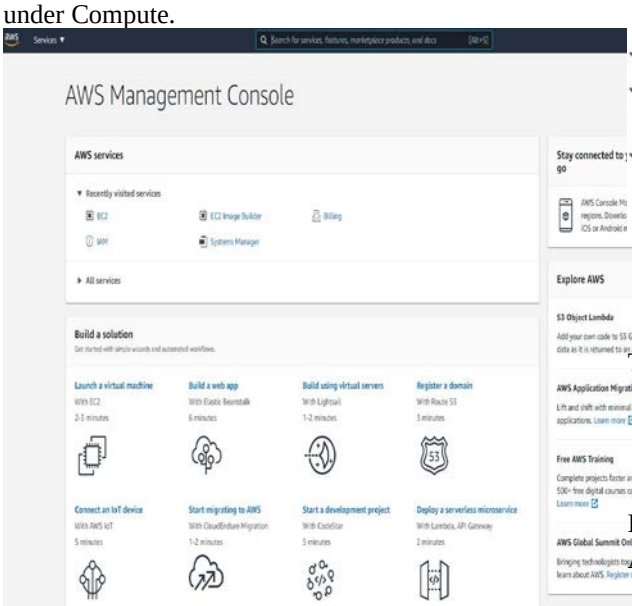
- Persevering storage capacities for our data using Amazon EBS
- Numerous physical locations for our resources (Regions and Availability Zones)
- A firewall that make us to clarify the protocols, ports, IP etc.
- Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses
- Virtual networks (VPCs) we can make for our instances that are logically isolated from the rest of the cloud.

III.III. System ArchItecture

As we mention earlier, we first need to show setting a new instance, a new virtual machine on a cloud service like AWS manually. For this purpose, let us show the steps and calculate how much time it will need to be to configure&set up a new instance to a hundreds of VMIs if we needed.

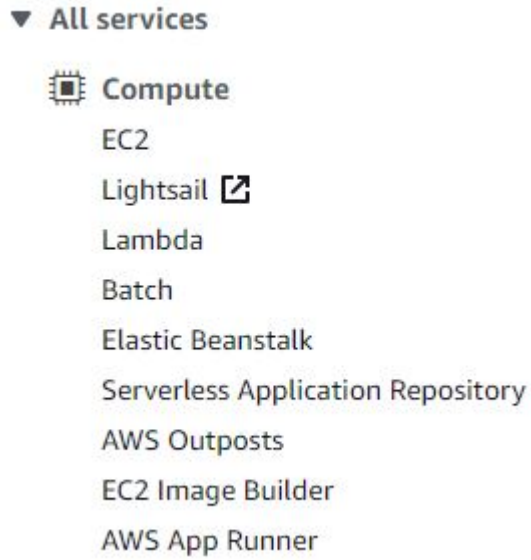
To launch an EC2 instance;

- Sign in to the AWS Management Console.
- Open the Amazon EC2 console by choosing EC2 under Compute.



- Figure 11 : General view of AWS Management Console

Choose all services;



After choosing EC2 as compute service,
If you are using the Show All Services view, your screen looks like this:

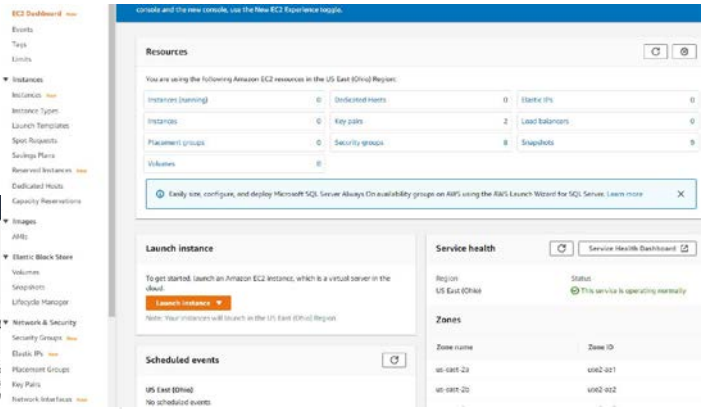


Figure 12 : EC2 Panel

There will be panel for;

- Instances
- Images
- Elastic Block Store
- Network&Security

Let's first focus on AMIs. We previously created all these AMIs with Packer only.

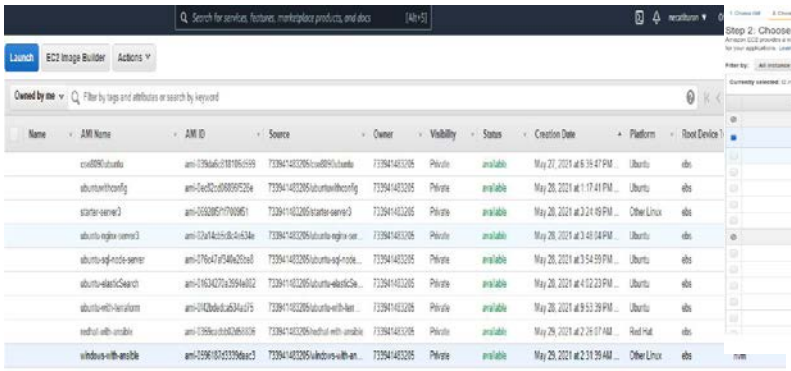


Figure 13 : AMIs Created By Packer

However this time it will be created manually by only using EC2 options. There will be 7 steps of AMI creation.
To make all configuration and set-ups done let's follow these steps carefully.

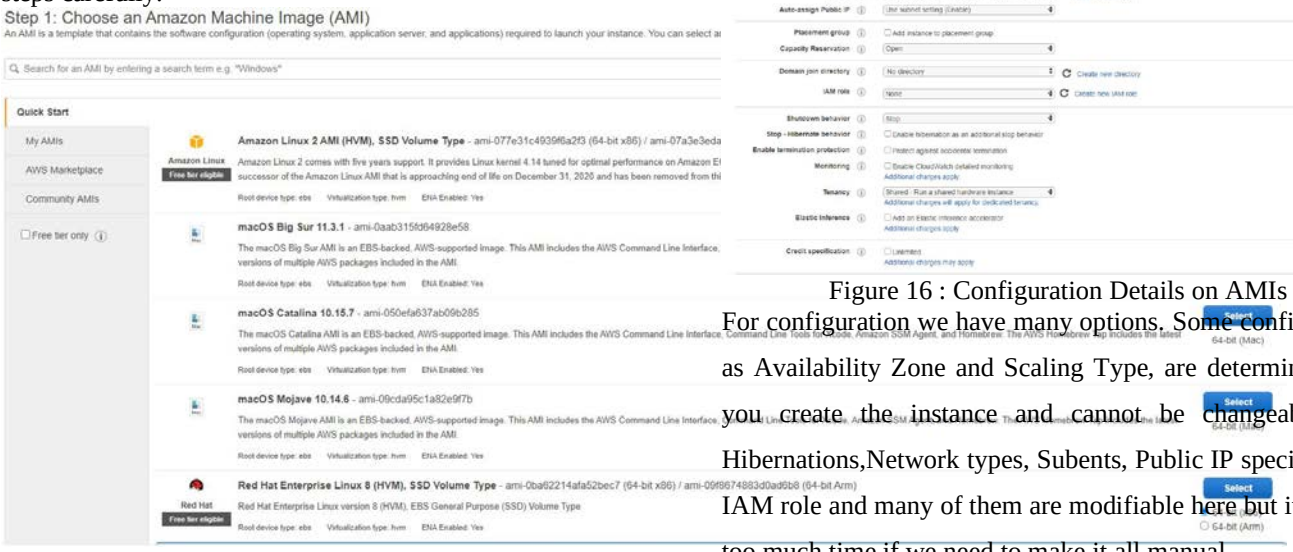


Figure 14 : First step of AMI Creation Manually

The AMI ID will be needed for creating instances of AMIs with Packer. However since it's now manual creation let's dive into Select option and choose an Amazon Linux AMI first.

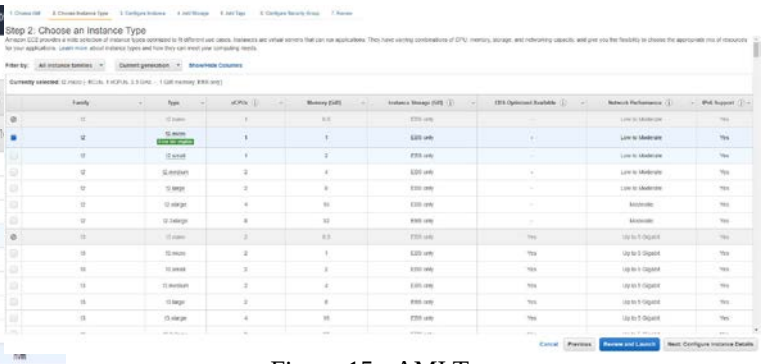


Figure 15 : AMI Type

Instance type is important to companies&services needs. This time we will continue choosing with t2.micro AMI instance.

Figure 16 : Configuration Details on AMIs

For configuration we have many options. Some configurations as Availability Zone and Scaling Type, are determined when you create the instance and cannot be changeable later. Hibernations,Network types, Subents, Public IP specifications, IAM role and many of them are modifiable here but it can take too much time if we need to make it all manual.

Figure 17 : Storage Details of AMI

We can add new volumes on our storage and make it special like, deletion after termination. We can also encrypt any part of it.

Tags are important if we need to make sure to specify one another.

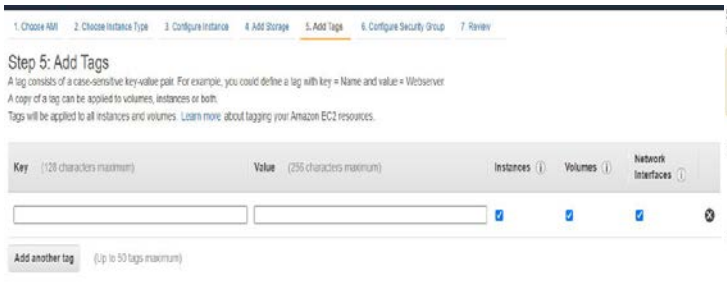


Figure 18 : Tags of AMI

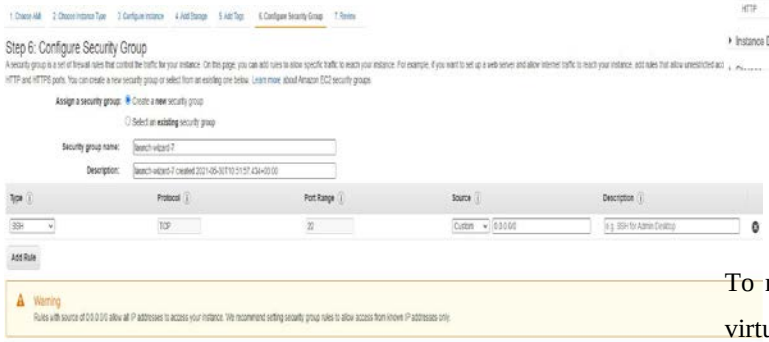


Figure 19 : Configuration fo Security Groups

Configuring security groups are crucial when it comes to delivery and availability. DevOPS teams must decide which types of rules (e.g.HTTP,HTTPS) to make AMI available from everywhere.

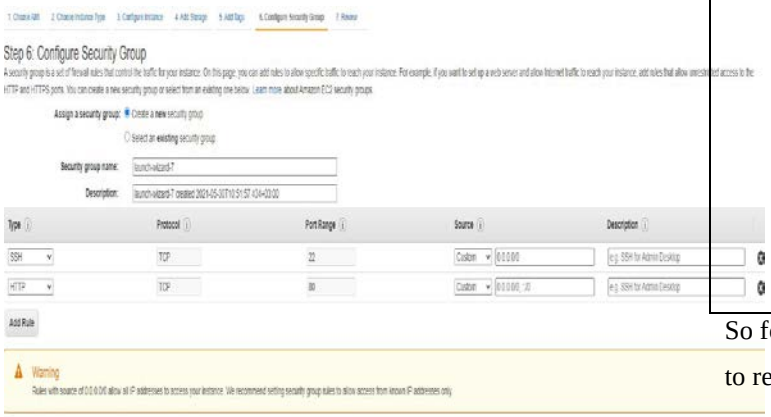


Figure 20 : Adding HTTP Rule

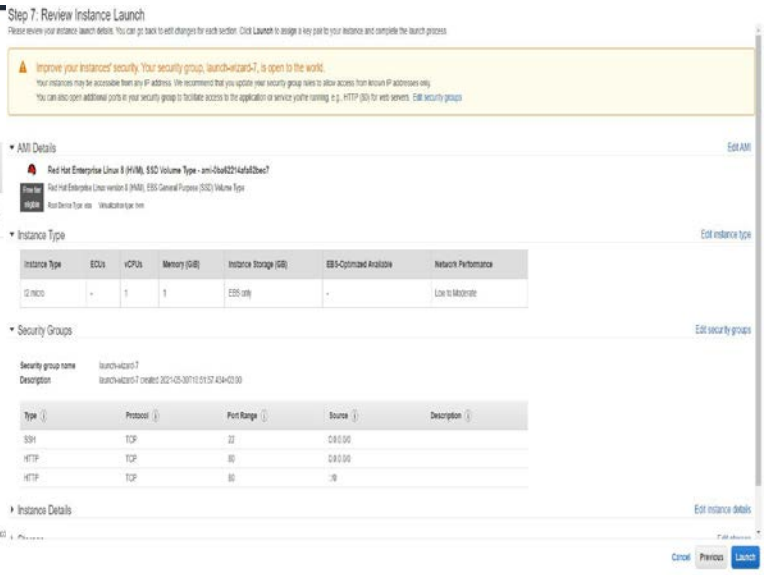


Figure 21 : Adding HTTPS Rule

To make our AMI reachable from other platforms and other virtual systems (e.g. vim,kernel,winRM) we must create a key pair. Key pair has a format of PEM (Privacy-Enhanced Mail) and it's a known standar for encryption certificates.

Its structures basics has this format :

-----BEGIN CERTIFICATE-----

(Your Primary SSL certificate: your_domain_name.crt)

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----

(Your Intermediate certificate: DigiCertCA.crt)

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----

(Your Root certificate: TrustedRoot.crt)

-----END CERTIFICATE-----

So for this purpose we need to download and hide a .pem file to reach our AMIs when it needed.

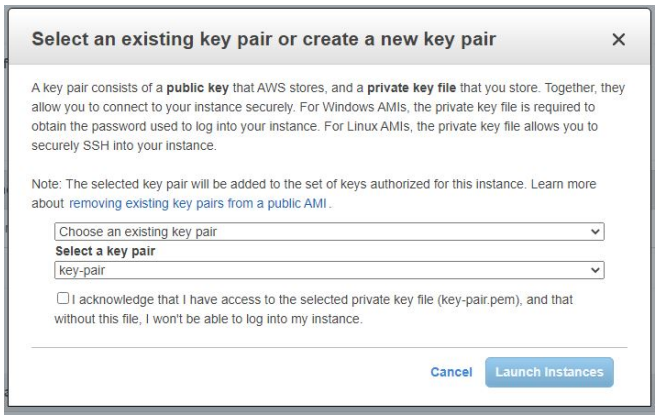


Figure 22 : Key pair (pem file)

After all this long and detailed steps we are close to end this creation of specific AMIs arranged for our purposes.

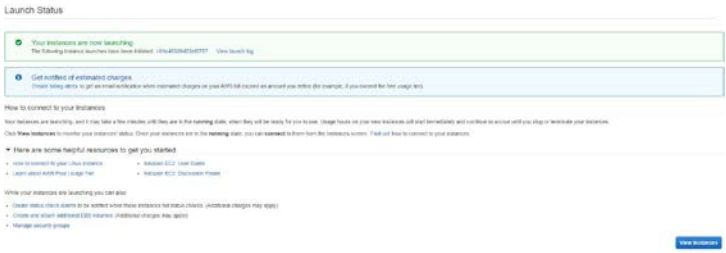


Figure 23 : End of the creation of AMI

Now in the instances panel, we can see our copy (instance of AMI) on the server.

As you can see these long steps can be quickly done by only Packer itself.

First, let's understand comprehensively Packer's hierarchy and components.

III.II. Module B

USING PACKER

Packer can be installed in the several ways:

- With a a pre-compiled bin. By using released binaries for all supported platforms and architectures.
- Installing from source code itself. (Can be useful if we only use shell commands rather than GUI)
- Using a system's package management tools (especially on linux deploys).

To use Packer and its components we have chosed Windows 10 as a base platform for configuring its specialities. Therefore we used Chocolatey as a helper tool for installing Packer itself.

To install chocolatey we must open PowerShell as administrative. After then we used shell commands as this ;

```
Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor
3072;
 iex ((New-Object System.Net.WebClient).
DownloadString('https://chocolatey.org/install.ps1'))
```

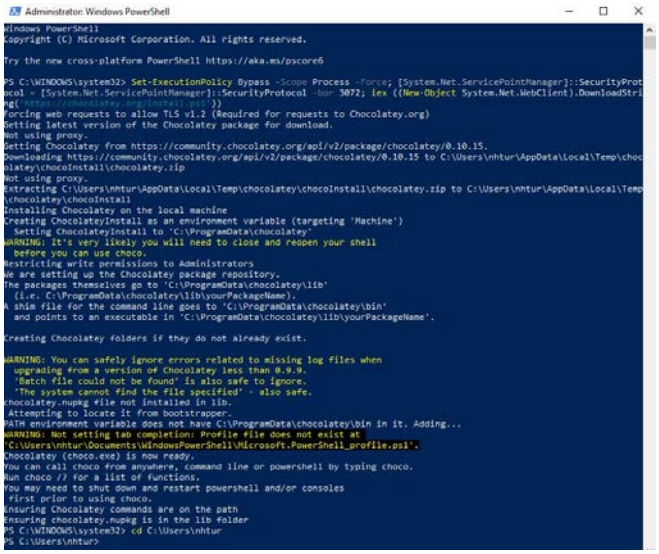


Figure 24 : Installing Chocolatey

Now the system is ready to install Packer with Chocolatey. In powershell, again we used a Chocolatey command.

```
$ choco install packer
```

After installation, a verification would be handy.

```
$ packer
usage: packer [--version] [--help] <command> [<args>]

Available commands are:
    build       build image(s) from template
    fix         fixes templates from old versions of packer
    inspect     see components of a template
    validate    check that a template is valid
    version     Prints the Packer version
```

Now we are ready to use all advantages of Packer. Before all of those, let us summarize what Packer have and how we can use its abilities.

Packer's Advantages

1) Continuous Delivery

Packer is small, portable, and command-line driven so it's easy to use remotely from any platform.
This makes Packer the perfect tool to put in the middle of a CD pipeline.
Packer also might be used to generate new VMIs for any kind of platforms on every change to compete Chef,Puppet,Ansible or for any other tools.
Belonging to the pipeline, the newest created VMIs might be instantiated,launched and tested after verifying the infrastructure mutations of the system's work.

2) Dev/Prod Parity

Packer helps keep all stages of the work; dev-staging-production all done in once.

Packer also might be used to generate hundreds of VMIs for any kind of platforms at the same time.

So if we use AWS for production but some other VMware for dev, we can generate both an AMI and a VMware image using Packer at the same time from same Packer template with a JSON file.

3) Appliance/Demo Creation

Since the tool creates consistent VMIs for more than once platform in parallel, it's the best tool for creating appliances and disposable product demos for our services.

As software changes, DevOPS teams can automatically create appliances with the software pre-installed.

This means that users can get started with new software by deploying it to the environment of their any desires.

Packer's Structure

Packer has 6 fundamental component in its hierarchy;

- Builders
- Provisioners
- Variables/Templates
- Paralel Builds
- Post-Processors
- Communicators

Using Packer To Create AWS AMIs

For the final part of the methodology section, we finally show how to use Packer to immediate creation of VMI instances.

System Tools & Specifications.

- VS Code
- Chocolatey
- Powershell
- Amazon AWS Account
- Windows 10 OS

Setting Up VS Code

After opening a new windows we created a new JSON file with a name as AWS. We no ready to create our first AMI from AWS.

For each platforms and services, we created a Packer JSON File as it can be seen on the figure.

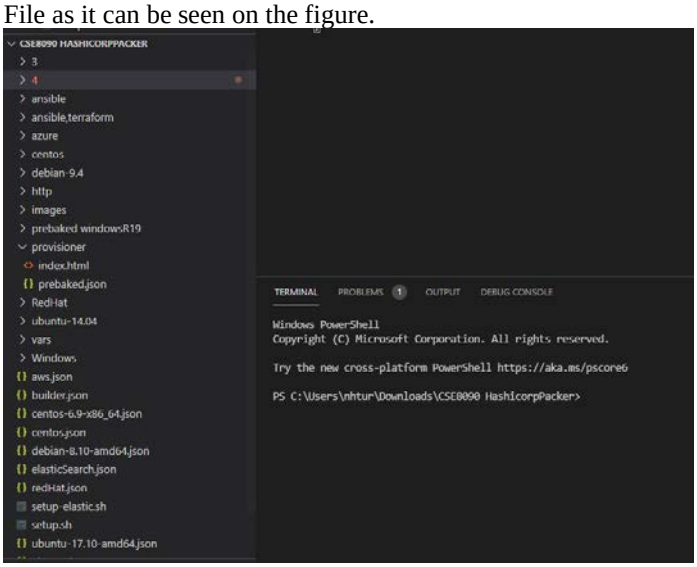


Figure 25 : Setting The VS Code for Packer

Every OS has different credentials, we must first check from AMAZON EC2 servers.

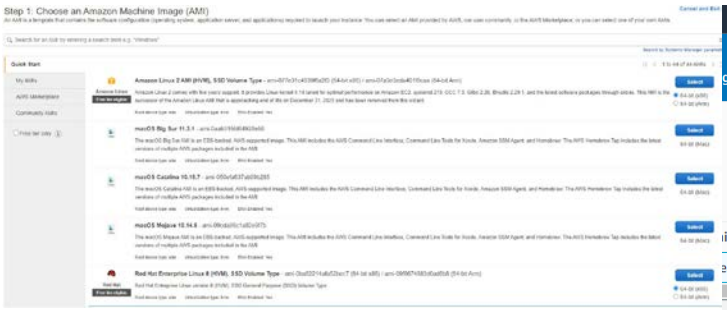
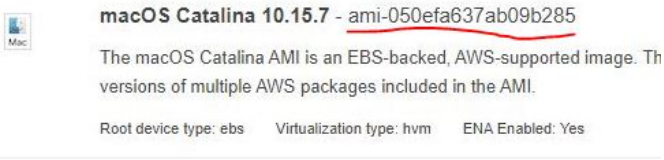


Figure 26 : Choosing AMI ID and Properties



Every AMI has different IDs and properties, we will need those credentials on the JSON file we will use.

First we start with a builder.

```
{
    "builders": [
        {
            "type": "amazon-ebs",
            "access_key":"AKIA2VYSSI3C52XY32R6",
            "secret_key":"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",
            "region": "us-east-2",
            "ami_name":"windows-with-ansible",
            "source_ami":"ami-00399ec92321828f5",
            "instance_type":"t2.micro",
            "ssh_username":"ubuntu"
        }
    ]
}
```

We need some components to build our first AMI ;

- Type: Which platform ?
- Access key: Specific key for each account
- Secret key: Secret key for each account
- AMI name : Can be chosen any name, no matter
- Source_ami : Specific ID of the AMI provided by Amazon AWS
- Instance_type: Which type of storage we need, previously showed in the 2nd section.
- Ssh_username:This is provided by amazon. Generally it's "root" or "ec2-user"

For access and secret keys we must take our secret keys from "My Security Credentials" as it seen from the figure.

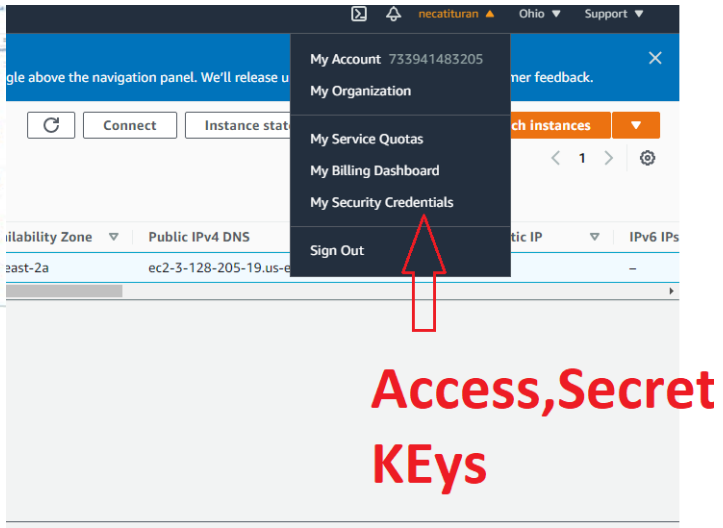


Figure 27: Security Credentials

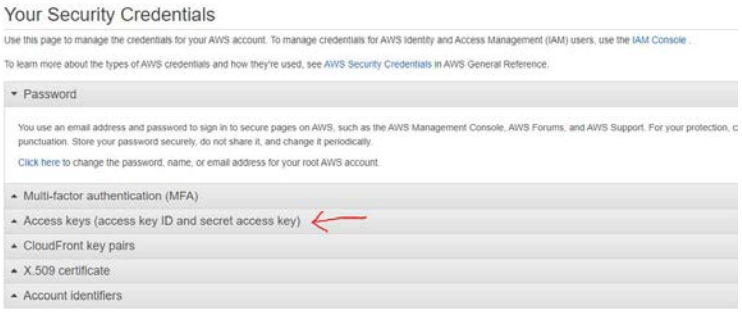


Figure 28 : Access Keys

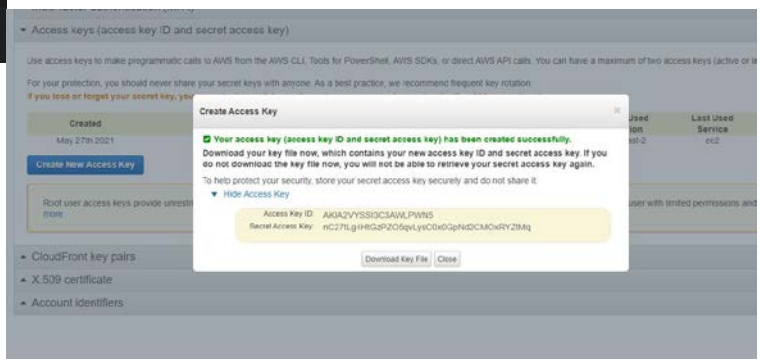


Figure 29 : Downloading Key

Now we can test our first AMI in the VS Code Terminal.
To build our AMI on the terminal, we use the basic command.

```
packer build test.json
```

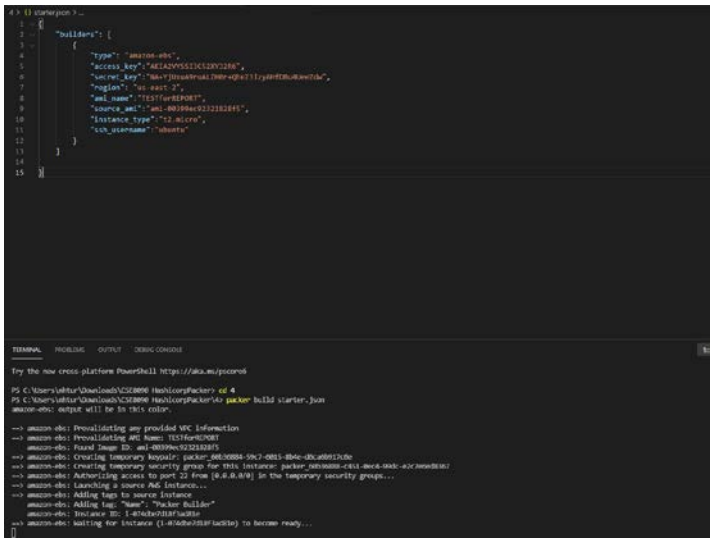


Figure 30 : Packer Build

Packer will create and AMI and for that purpose it will create an instance and then it will stopped that immediately after creating.

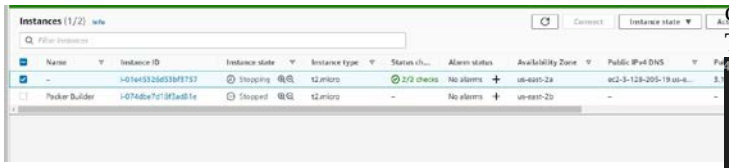


Figure 31 : Packer Builder Seen on AWS

AMI successfully created in 25 seconds from terminal by emote access using only Access keys and secret keys.

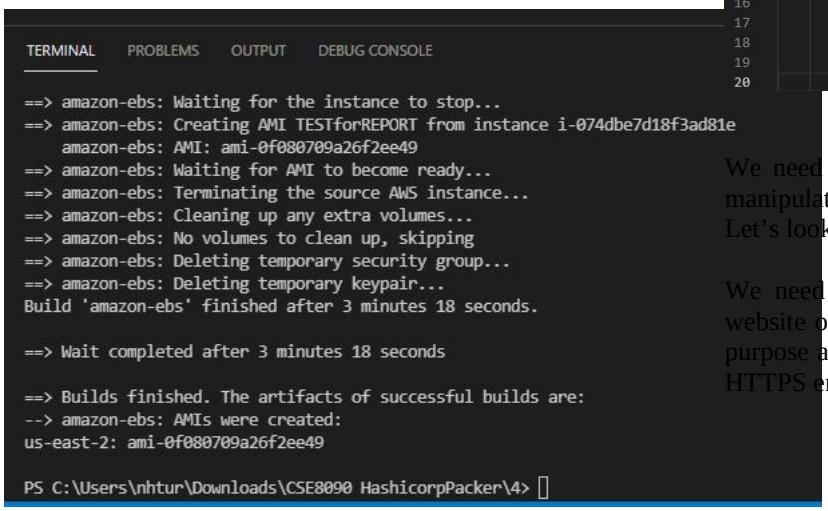


Figure 32 : AMI Created

AMI now can be seen on the EC2 console panel's AMI section.

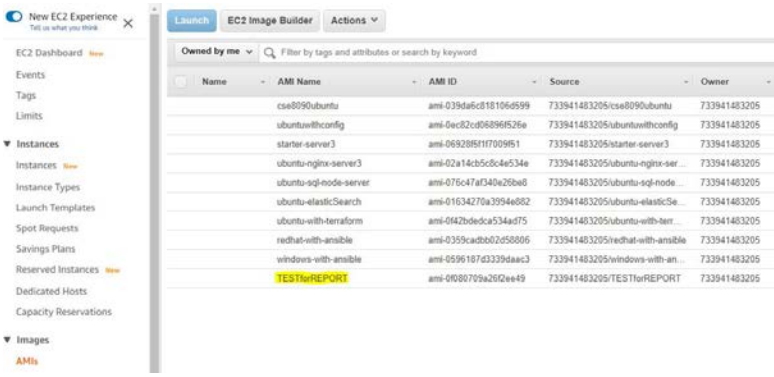


Figure 33 : AMI Seen on AWS

Now we can go further and by only using Shell commands, we can create a directory for our WebSite in the AWS cloud.

We can manipulate all the website and its packages from our JSON File.
Let's go step by step ;
First again we will build our builder by specific details of the chosen AMI (which is ubuntu for this example)
To make configurations, we need a set of Provisioners.

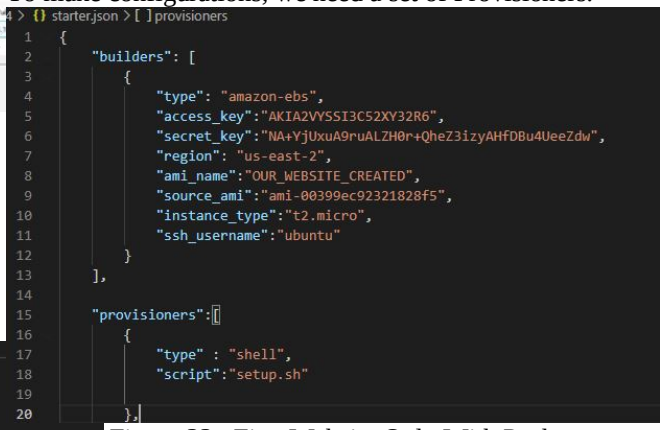


Figure 33 : First Website Only With Packer

We need shell and script provisioner to create a .SH file to manipulate the AMI.
Let's look at first .SH File.

We need NGINX server to be configured first to create a website on AWS. We also need to update the system for this purpose and to make it publicly reachable we need HTTP and HTTPS enabling.

```
setup.sh 4 X  {} redHat.json  {} prebaked.json  {}
4 > setup.sh
1   sleep 30
2   sudo apt update
3   sudo apt install nginx -y
4   systemctl enable nginx
5   sudo ufw allow ssh
6   sudo ufw allow http
7   sudo ufw allow https
8   sudo ufw enable
9
```

Figure 34 : Shell Commands for NGINX Server

After creating the setup.SH file, we need to attach and make it run in AWS by using packer's provisioners.

```
setup.sh 4 ●  {} starter.json X  {} redHat.json  {} prebaked.json  {} aws.json
4 > {} starter.json > [ ] provisioners
6               "secret_key":"NA+YjUxuA9ruALZH0r+QheZ3izyAHtDBu4UeeZdw",
7               "region": "us-east-2",
8               "ami_name":"OUR_WEBSITE_CREATED",
9               "source_ami":"ami-00399ec92321828f5",
10              "instance_type":"t2.micro",
11              "ssh_username":"ubuntu"
12          }
13      ],
14
15      "provisioners":[
16          {
17              "type" : "shell",
18              "script":"setup.sh"
19
20          },
21          {
22              "type":"file",
23              "source":"index.html",
24              "destination":"/tmp/"
25          },
26
27          {
28              "type":"shell",
29              "inline":["sudo cp /tmp/index.html /var/www/html/"]
30          }
31
32      ]
33  }
```

Figure 35 : Sudo Copy Directory

We copy our index.html file to the server by using Sudo privilege from var/www/ to tmp directory.

```
setup.sh 4 ●  {} starter.json  {} redHat.json  {} prebaked.json  {} aws.json  {}
4 > index.html > html
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>CSE8090 Graduation Project</title>
8   </head>
9   <body>
10      <h1>Project Owners</h1>
11      <h3>Necati H.Turan</h3>
12      <h3>Işıl Alıcı</h3>
13      <h3>Yaren Mısırlı</h3>
14
15  </body>
16  </html>
```

Figure 36 : Simple HTML File with NGINX

Here is our AMIs public IP address, it's running on the our AWS account.

Instances (1/3) Info

Filter instances

Name	Instance ID	Instance state	Instance type
–	i-01e45326d53bf3757	Stopped	t2.micro
Packer Builder	i-074dbe7d18f3ad81e	Terminated	t2.micro
–	i-0d6e0a4f9d4960f22	Running	t2.micro

Figure 37 : Public IP

We can now reach our website with the IP address from anywhere.

18.222.78.158

Project Owners

Necati H.Turan

Işıl Alıcı

Yaren Mısırlı

Figure 38 : Up and Running The Website

From now on, we can manipulate our services and kernell's feature remotely by only using PowerShell or from any platform we need.

By using our SSH, writed by only key pair direction and the our IP address ;

```
Ssh -i C:\Users\nhtur\Downloads\key-pair.pem
```

ubuntu@18.222.78.158

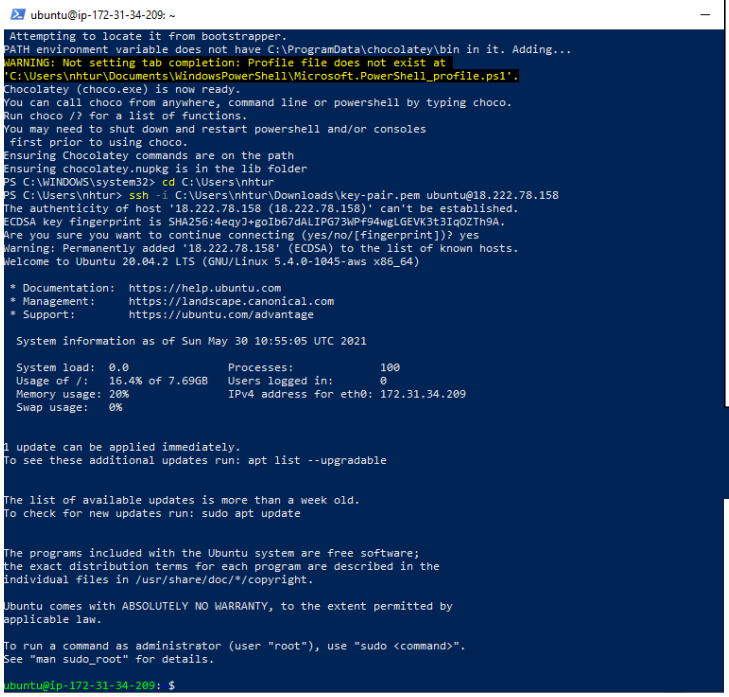


Figure 39 : SSH Reach

We can change and configure privileges by using SUDO or any platform specific command-line helpers.
Here is a list of readable, writable and other permissions of files.

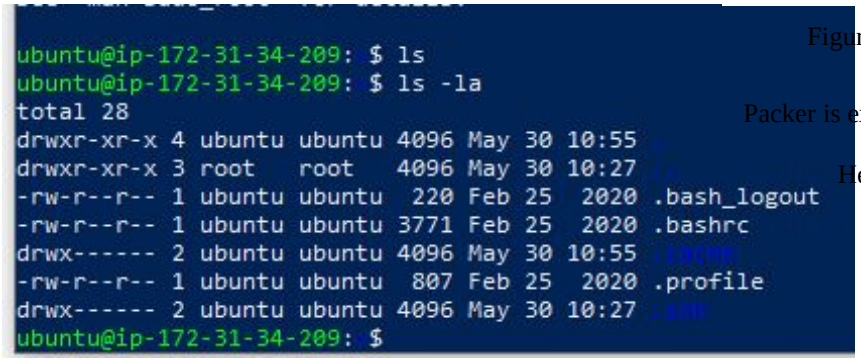


Figure 40 : DRWXR

IV. Experimental Studies

After our work done with Packer, to prove our points and show comprehensive comparisons on Packer, we aim to show studies from other works and statistics about Packer's opponents.
We showed Packer's abilities and Immutable servers capabilities.

Here is a resulted diagram of the Immutable again.

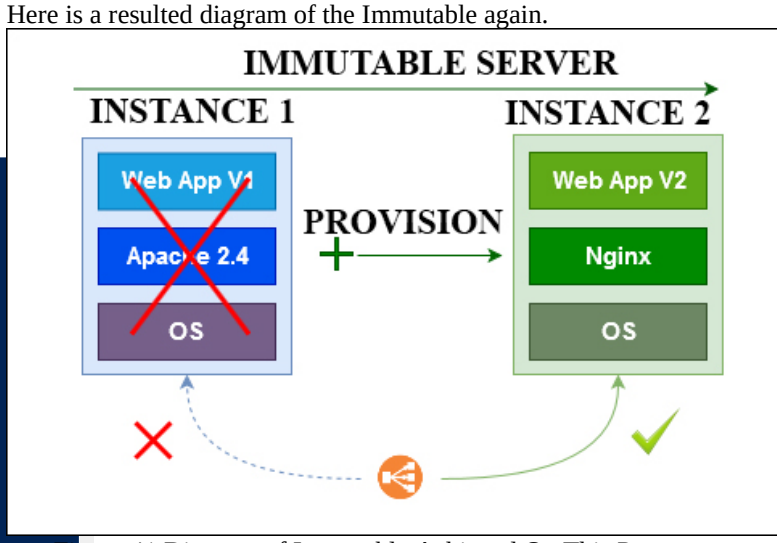


Figure 41:Diagram of Immutable Achieved On This Paper

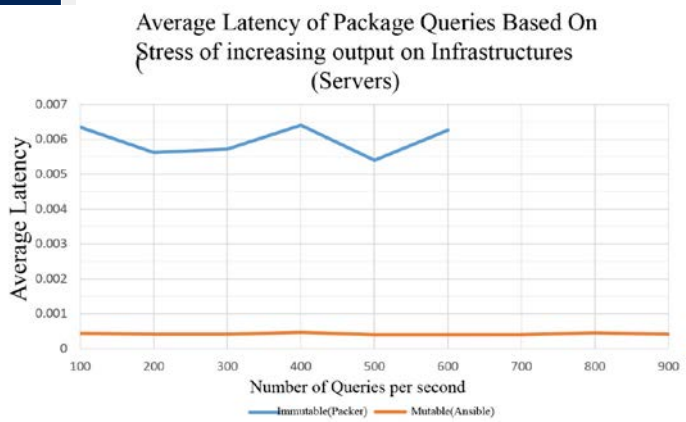


Figure 42 : Benchmark of Packer vs. Opponent

Packer is exceeds over its opponents by memory usages on cloud services like EC2,Azure.
Here is a figure to show this passage[14].

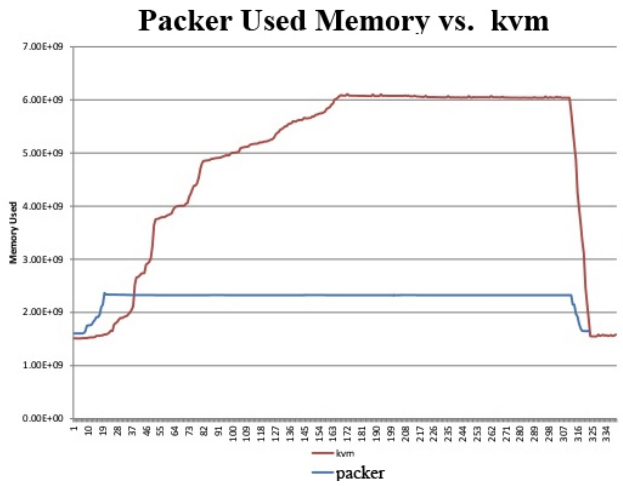


Figure 43:Packer vs. Opponent on Memory Usage

There is also a study on UBUNTU, to prove immutable works faster especially on large scale apps like Notion (notetaking app)[11][13].

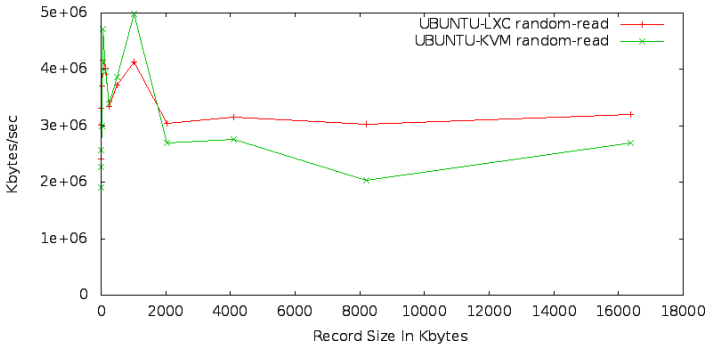


Figure 44: Ubuntu on Cloud Record Sizes

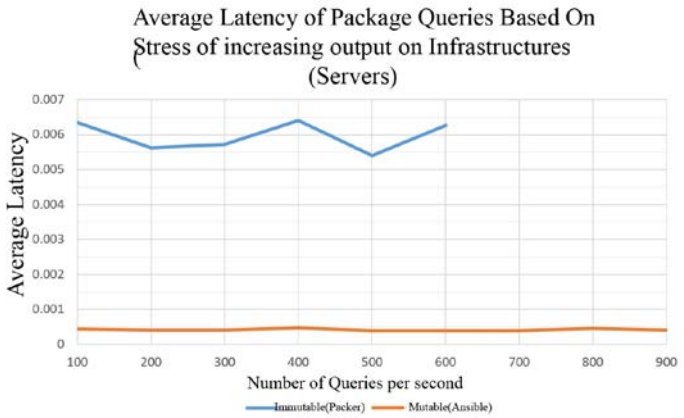


Figure 42 : Benchmark of Packer vs. Opponent

Packer is exceeds over its opponents by memory usages on cloud services like EC2,Azure.

Here is a figure to show this passage[14].

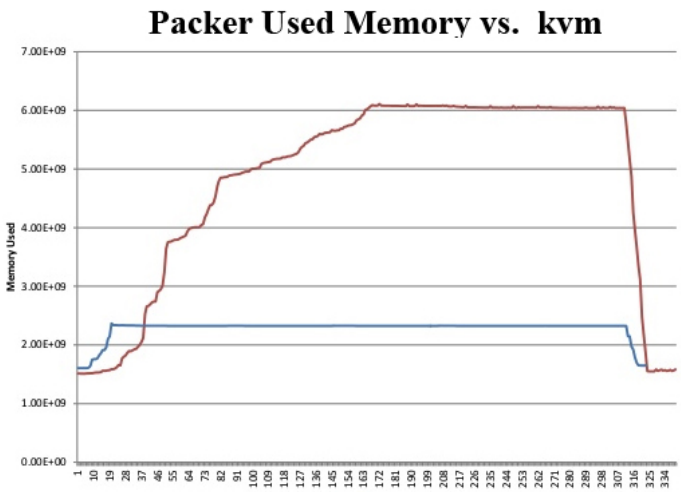
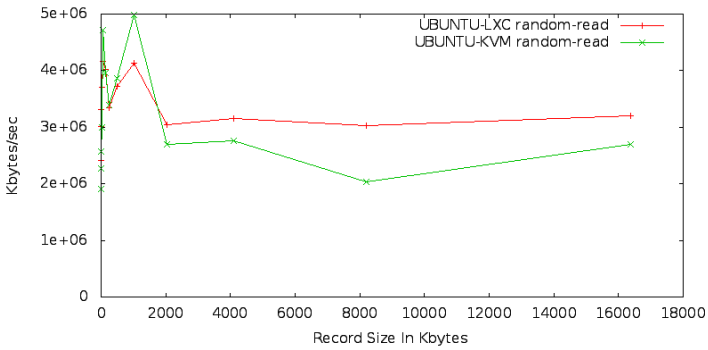


Figure 43:Packer vs. Opponent on Memory Usage

There is also a study on UBUNTU, to prove immutable works faster especially on large scale apps like Notion (notetaking app)[11][13].



A. Abbreviations and Acronyms

Define VMI: Virtual Machine Image

JSON : JavaScript Object Notation

VPC : Virtual private cloud

IaC : Infrastructure as a Code.

IaaS : Infrastructure as a Service

PaaS : Platform as a Service

AWS : Amazon Web Services

AMI : Amazon Machine Image

SUDO : Super User Do

Ec2 : Elastic Cloud Compute

IEEE: Institute of Electrical and Electronics Engineer

HCL : Hashicorp Configuration Language

GAN:Generative adversarial networks

NAS:Network attached storage

API: Application Programming Interface

GPIO: General Purpose Input/Output RAM:Random access memory

SSH: Secure Shell OS:Operating system

K8S : Kubernetes

CaaS : Container as a Service.

ASNaC :Application-Specific Network-as-code

CD : Continuous Delivery

CI : Continuous Integration

AMAZON EBS : Amazon Elastic Block Store

VPCs : Virtual Private Clouds

- The abbreviation “i.e.” means “that is,” and the abbreviation “e.g.” means “for example.”

An excellent style manual for science writers is [7].

V. DISCUSSION

In The Automated Generation of Virtual Machine Images with HasiCorp Packer project aimed the change the notion of using infrastructure of the software deployment and tried to reduce significant amount of time and cost of the general develop-configure-deploy phase of the systems that happene to be installed on local deviceso or cloud based systems like AWS,Azure or Digital Ocean. With these concept of the deployment we automated the system of creation of the VMs and its configurations with the each phases of the required cycle of the generations VMs and secondary/third party applications.

VI. CONCLUSIONS

For developing the most effective deployment system for our services and applications we defined the mutable and immutable systems.

Immutable systems are knowable due to the fact that they have no alterations in it. With our basic introduction of Immutable Infrastructures tried to make a corresponding shift between Immutable infrastructure, is enlight the fact it instantiate a better ,more efficient and most importantly a faster provided desired whole system and we used for this purpose the Hashicorp’s ecosystem’s Packer. We have also have discussed the packer’s abilities and benefits when it comes to comparison of the other DevOPS tools like Linkit, Ansible and Terraform. To make our points solid, this paper tried to cover all the issues and uses of a infrastructure by using Packer itself and by showing other techniques used to be popular. We also talked decently about DevOps maintenance and configurated systems of their own evolution. The main purpose and the main contribution of this paper is to enlighten reader about the approach of immutable infrastructure among other techniques by using Packer and comparing others with it.

REFERENCES

[1] L. J. Bass, I. M. Weber, and L. Zhu, DevOps - A Software Architect's Perspective., ser. SEI series in software engineering. AddisonWesley, 2015.

[2] K. Morris, Infrastructure as Code. O'Reilly Media, Inc., 2016. [Online]. Available: https://www.oreilly.com/library/ view/infrastructure-as-code/9781491924334/

[3] P. Lipton, D. Palma, M. Rutkowski, and D. A. Tamburri, "TOSCA Solves Big Problems in the Cloud and Beyond!" IEEE Cloud Computing, pp. 1–1, 2018.

[4] Within-project Defect Prediction of Infrastructure-as-Code Using Product and Process Metrics Stefano Dalla Palma∗ , Dario Di Nucci∗ , Fabio Palomba† , and Damian A. Tamburri‡ ∗Tilburg University, Jheronimous Academy of Data Science, Netherlands †Software Engineering (SeSa) Lab — University of Salerno, Italy ‡Eindhoven University of Technology, Jheronimous Academy of Data Science, Netherlands

[5] Managing Virtual Appliance Lifecycle in IaaS and PaaS Clouds, Michal Kimle∗ , L'ubomír Košarišt'an, Boris Parák, Zdenek Šustr

[6] On the Effectiveness of Tools to Support Infrastructure as Code: Model-Driven Versus Code-Centric, Julio Sandobalin, Emilio Insfran

[7] Peter Vaillancourt, Bennett Wineholt, Brandon Barker, Plato Deliyannis, Jackie Zheng, Akshay Suresh, Adam Brazier, Rich Knepper, and Rich Wolski. 2020. Reproducible and Portable Workflows for Scientific Computing and HPC in the Cloud. DOI:https://doi.org/10.1145/3311790.3396659

[8] Jack Cook, Richard Weiss, and Jens Mache. 2020. Refactoring a full stack web application to remove barriers for student developers and to add customization for instructors. J. Comput. Sci. Coll. 36, 1 (October 2020), 35–44.

[9] Elena A. Araujo, Álvaro M. Espíndola, Vinicius Cardoso Garcia, and Ricardo Terra. 2020. Applying a Multi-platform Architectural Conformance Solution in a Real-world Microservice-based System. In Proceedings of the 14th Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS '20). Association for Computing Machinery, New York, NY, USA, 41–50. DOI:https://doi.org/10.1145/3425269.3425270

[10] A. Cepuc, R. Botez, O. Craciun, I. -A. Ivanciu and V. Dobrota, "Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins, Ansible and Kubernetes," 2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet), 2020, pp. 1-6, doi: 10.1109/RoEduNet51892.2020.9324857.

[11] Managing Virtual Appliance Lifecycle in IaaS and PaaS Clouds, , L'ubomír Košarišt'an, Boris Parák, Zdenek Šustr ˇ

[12] NIFTY, [Online] Available: https://github.com/CESNET/nifty [Accessed: November 11, 2016].

[13] ITCHY, [Online] Available: https://github.com/CESNET/itchy [Accessed: November 11, 2016].

[14]] HEPIX Virtualisation Working Group, August 26, 2012, [Online] Available: http://grid.desy.de/vm/hepix/vwg/doc/pdf/Book-a4.pdf [Accessed: November 11, 2016].

[15] Docker, [Online] Available: https://www.docker.com/ [Accessed: November 11, 2016].

[16] git-scp, [Online] Available: https://git-scm.com/ [Accessed: November 11, 2016].

[17] cloud-init, [Online] Available: https://cloudinit.readthedocs.org/en/latest/ [Accessed: November 11, 2016].

[18] COMFY, [Online] Available: https://github.com/CESNET/comfy [Accessed: November 11, 2016].

[19] Y. Brikman, Terraform: Up and Running, 1st ed. Newton, MA, USA: O'Reilly Media, 2017.

[20] Y. Martínez, C. Cachero, and S. Meliá, ''MDD vs. traditional software development: A practitioner's subjective perspective,'' Inf. Softw. Technol., vol. 55, no. 2, pp. 189–200, Feb. 2013

[21] Amazon Web Services. (2011). OpsWorks. Accessed: Aug. 29, 2019. [Online]. Available: https://aws.amazon.com/opsworks/

[22] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, ''A systematic mapping study of infrastructure as code research,'' Inf. Softw. Technol., vol. 108, pp. 65–77, Apr. 2019

[23]

[24] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[25] I.S. Jacobs and C.P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

[26] K. Elissa, "Title of paper if known," unpublished.

[27] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[28] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[29] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.