ISTANBUL KÜLTÜR UNIVERSITY

Graduation Project
Automated Generation of Virtual Machine Images with Hasicorp Packer

Submitted By

Necati Hasan Turan 1101020051
Yaren Mısırlı 1600001547
Işıl Alıcı 1600002106

Project Advisor
Prof.Dr. Murat Taylı

Department of Computer Engineering
İstanbul Kültür University

2020-2021 SPRING

Automated Generation of VMIs with Hasicorp Packer

May
2021

ISTANBUL KÜLTÜR UNIVERSITY

Graduation Project
Automated Generation of Virtual Machine Images with Hasicorp Packer

Submitted By

Necati Hasan Turan 1101020051
Yaren Mısırlı 1600001547
Işıl Alıcı 1600002106

Advisor
Prof.Dr. Murat Taylı

THE EXAMINATION COMMITTEE

Jury Member Signature

1. Prof.Dr.Murat TAYLI ………………………..
2. Prof.Dr.Akhan AKBULUT ………………………..
3. Prof.Dr.Özgür Koray ŞAHİNGÖZ ………………………..

ABSTRACT

Modern software development depends on the benefits of mechanization on development and operational stages, an engineering concept called as DevOps (development&operations).

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 create a copy of running operating systems as 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.

ÖZET

Modern yazılım geliştirme DevOps olarak bilinen geliştirme ve operasyonel süreçlerin otomasyon halinde planlamasıyla oluşan mühendislik konseptine dayanmaktadır.

Bulut bilişim çağında, birçok kuruluş monolitik sistemleri bırakıyor ve geliştirme-yapılandırma-dağıtma aşamalarını Kod olarak Altyapı kavramına kadar güvenilirliğini basitleştirmek için altyapılarını ***değiştirilemez*** altyapıya dönüştürüyorlar. Genel geliştirme, yapılandırma ve dağıtma aşamaları artık her zamankinden daha karmaşık, daha fazla zaman ve maliyet gerektiriyor. Hizmetlerimiz ve uygulamalarımız için yapılandırılmış bir sunucu ve platform oluşturmak için geliştirdiğimiz ve hizmet verdiğimiz platformlarda ***sanal bir makine görüntüsü*** kullanıyoruz. Bir makine görüntüsü, önceden hazırlanmış ve ayarları tamamlanmış bir işletim sisteminin sanal bir kopyasıdır. Bu görüntü içerisinde, istediğimiz amaçlarla hızlı bir şekilde yeni çalışan makineler oluşturmak için kullanılan yazılımlar, yamalar, ağ yapılandırması yüklenmiş olması gerekir. Önceden pişirilmiş görüntülerin kullanılması, etkinlik ve tutarlılık sağlar ve geliştirme sürelerini diğer tekniklerden önemli ölçüde azaltır. Ancak makine görüntü türleri ve yapılandırmaları her platform için değişir. Bu, her değişikliği, yapılandırmayı ve güncellemeyi uygulamalarımızda hizmet vereceğimiz her platform için uyarlamamız gerektiği anlamına gelir. Değişmez bir altyapı ile çalışan bir sunucuda güncelleme yapmak yerine her seferinde yeni bir sunucu oluşturuyoruz. Değişmez bir altyapıyı uygulamanın ve kullanmanın en verimli yolu, sistemin her parçasını tek bir kod birimi yapmaktır. Bu nedenle, sanal sunucu görüntüleri, sunucunun belirli değişiklikleri gibi gerekli tüm parçaları içeren ***Kod olarak Altyapı*** terimini kullanıyoruz. Tüm bu platformları ve yapılandırmalarını oluşturmak, geliştiriciler için çok karmaşık ve zaman ve maliyet alıcı bir aşamadır. Bu sorunu çözebilmek için Değişmez Altyapı oluştururken Hashicorp Packer'ın tüm yeteneklerini kullanmayı amaçladık. Herhangi bir platformdan yalnızca JSON formatlama dosyasıyla önceden pişirilmiş makine görüntüleri oluşturarak bunu çözmenin hızlı ve iyi bir yolunu bu araştırmada uyguladık ve kanıtladık. Packer, her türden makine görüntüsünün oluşturulmasını otomatik hale getirir ve uzaktan çalışmasını sağlar. Packer ile oluşturulan resimlerinizde yazılımı kurmak ve yapılandırmak için Chef veya Puppet gibi bir çerçeve kullanmaya teşvik ederek modern konfigürasyon yönetimini içerir.

ACKNOWLEDGEMENTS

Firstly, we would like to thank Istanbul Kultur University Computer Engineering department for their contributions and support through the years we educated and for the preparation of this project and report.

We would like thank all of the participants of the project for their valuable efforts.

Finally we would also like to express our great gratitudes and best wishes Prof. Dr. Murat TAYLI for his mentorship, experience, guidance and care through the project and making this project happened.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS & ABBREVIATIONS

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

1. 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;

1. Infrastructure as a Service -IaaS-,
2. Software as a Service -SaaS-,
3. Platform as a Service -PaaS-

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.

1.1. Problem Statement

When we try to deploy a series of apps, configurations and services for a certain aim on an instance of a virtual image, it will take longer time to make it ready for work and do exact setting of our desired purpose.

We, developers always want to automate the system so DevOps teams can easily maintain and alter each time we need to be interact with our servers.[5]

On DevOps[1]; software codes emerging as an automation is called as Infrastructure-as-Code (IaC). Infrastructure-as-Code means; promoting managing the information and late works inside re-usable scripts of IaC rather than old-school reserving it for the vast amount of work for developers, which makes the process quite slow, time and money wasting, and generally open to make errors [2].

Traditional mutable server infrastructure needs to be continually updated and altered for each configurations. That means developers must work with this the infrastructure and make secure shell into their servers -either upgrade or downgrade- packages one by one manually, change configuration files on a server by working for each (an infrustructure can have hundereds of servers) and deploy new set of code onto existing servers. Whole these configurations means that this servers servers are changeable (mutable) so they can be altered after they’re created. [7]

8

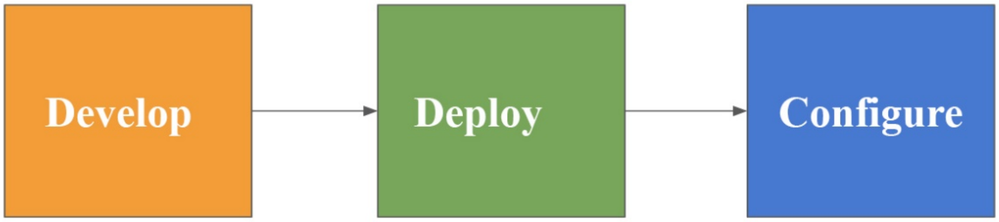


Figure 1 : Mutable / Normal Flow

However we need to eliminate this process and make autonomous. Therefore we resort to the Immutable Infrustructures which means kill the server if you want to make a change. To reduce the time for creating and configuring machine images we need to develop complex infrastructure and modify each time whenever we needs an update.

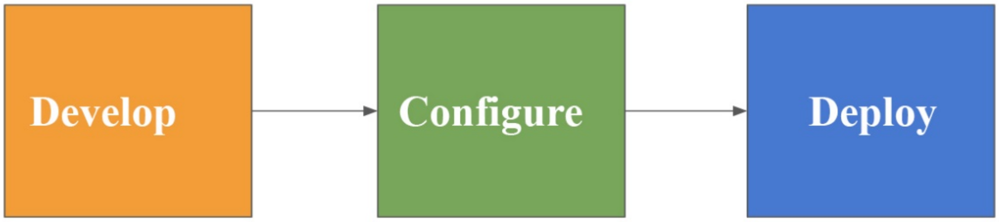


Figure 2 : Immutable Flow

We used Immutable Infrustructure There it's a problem not to have a tool like Packer.

1.2. Project Purpose

This project aims to show every capability of Packer by creating and launching machine images on AWS EC2 platform while showing how long it can take without packer.

While we show other techniques and their capacities, we will also show how easy it could with only using Packer as a helping tool to organize whole infrastructure for our services and applications.

We used Immutable Infrustructure[7] to prove our point we compare with other concepts like Mutable Infrustructure or monolith systems.

1.3. Project Scope

DevOps is an collective&extensive concept which reduces the blockage between development teams and operational staff to make a continuous and fast delivery and allow precise&fast reactions to evolving necessities in the infrastructure development cycles.[3]

Developing applications, running them and their applications needs to be established as IaaS,PaaS or as Monolithic traditional service.

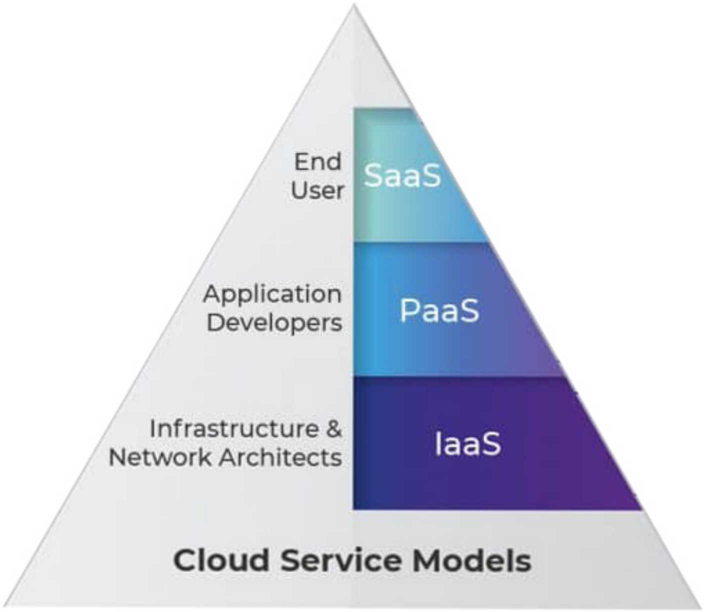


figure 3 :Cloud Service Models

Cloud providers offers a different kind of infrastructure for each purposes, the meaning of an infrastructure component (e.g., a virtual machine) implies creating a small amount of code that only focus on the aimed provider as cloud service prodivder.[10]

In this project, an automated machine images generation will be implemented by using Hashicorp Packer. We focus on the cloud concept as IaC. IaC approach supports code-oriented tools that use only scripts to specify configuring, creating, updating and finally executing the cloud infrastructure resources (e.g. VMIs) and their appliances (e.g. 3rd party applications, network configurations). [4][2]

We will discuss why IaC concept is necessary, what else we have, what can we do about solving popular industrial problems and finally we will show how we can achieve all of this with only Packer.

Project will show each parts&components of Packer and their applications on different operating systems and cloud based services[5].

1.4. Objectives and Success Criteria of the Project

The primary goal of the success of the project is to create a problemless generation of automated instances of virtual images and needed configurations in it. After the desired configurations and alterations is achieved on the our instance, the goal is to deal with in the project will be to have creation process only by a JSON file. The manual creation of the VMIs will be saved and showed in the project first.[6]

This recording will be showed on the AWS EC2 platforms. Then, with the Packer's abilities and components we will look at how we can do all of these manual configuration with an only a JSON file remotely. We don't even have be on the EC2 console or any platform we want to create our instances. Thus, the process of the generation of the automated virtual images will be done by our simple code blocks with the open source tool, Hashicorp Packer.

1.5. Report Outline

First, we will talk about virtual images, immutable and mutable infrustructures and then we will talk about types of virtual images. Then we will give a basic report about Amazon AWS ecosystem. Then we will talk about Hashicorp ecosystem. Finally we will take look at Packer, it's capabilities and components. In the methodology section, all parts of the our whole work and the Packer itself will be completely explained and showed. On the part of 5, the project with similar works has been examined and the evaluations are presented in a table.

In the experimental results section, the studies and observations made to achieve the targeted results are included. The data obtained as a result of these experiments are presented below with figures and tables.[16] In the discussion part, the procedures and findings obtained from the experiments were evaluated and the ideas were given. In the conclusion part, a short description of the project and future ideas are given.

2. RELATED WORK

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 set the basic principle of infrastructure requirements for that software such as Amazon cloud, Azure and the other numerous kind of VMIs needed in each fields.[23]

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.

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***.

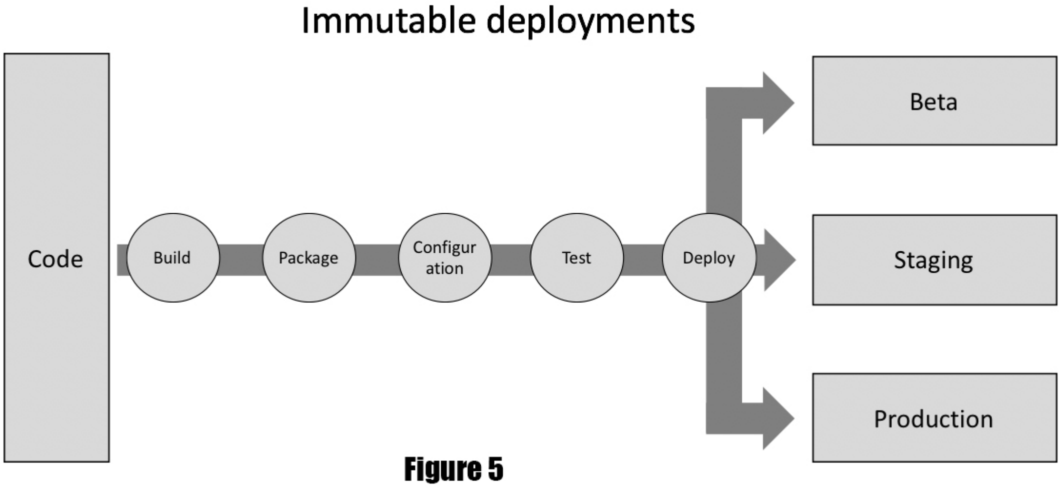


Figure 5

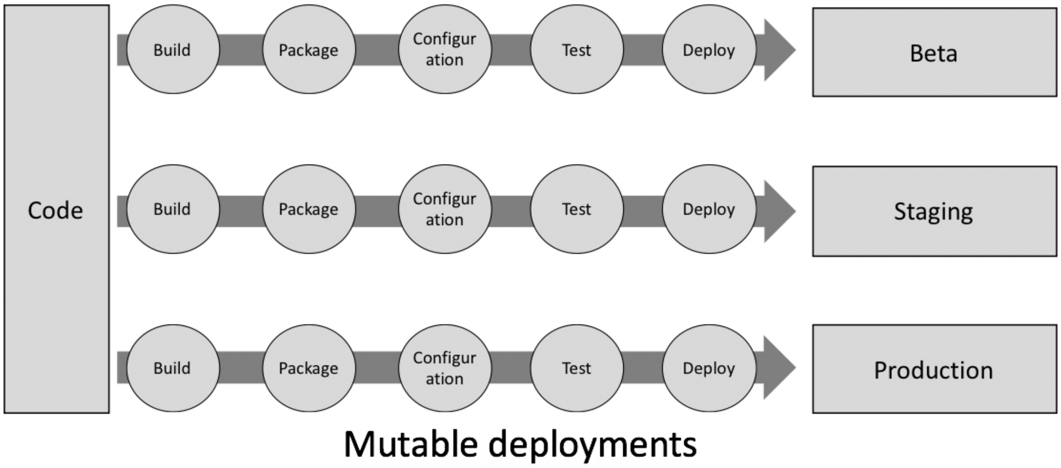
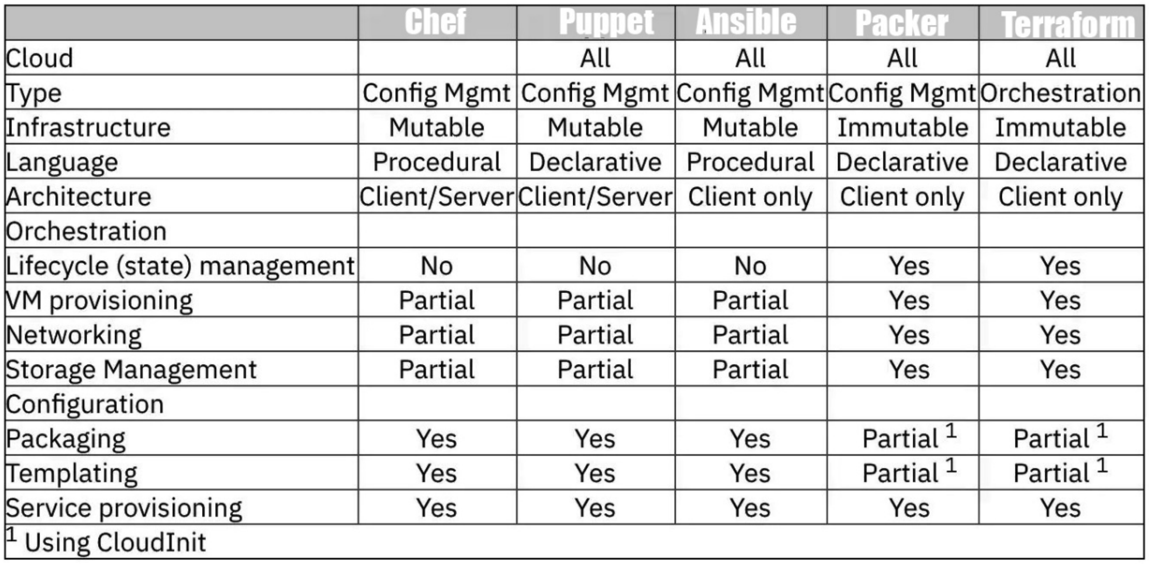


Figure 6

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.[22]

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

2.1. 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.

Generally, in mutable infrastructure DevOps 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 infrastructures.

Therefore immutable infrastructure is better for consistency.

2.1.1.Mutable

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 Linux :

```
$ sudo apt-get upgrade
$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo apt-get check
$ sudo apt-add-repository --yes --update ppa:ansible/ansible
$ sudo apt-get install ansible
```

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:

```
- name: Update the servers
  hosts: webservers
  remote_user: root

  tasks:
  - name: Making it sure apache is at the latest ver.
    ansible.builtin.yum:
      name: httpd
      state: latest
  - name: Write an apache configuration file
    ansible.builtin.template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- name: Update database server
  hosts: databases
  remote_user: root

  tasks:
  - name: Ensure postgresql is at the current ver.
    ansible.builtin.yum:
      name: postgresql
      state: latest
  - name: Ensure that postgresql is started
    ansible.builtin.service:
      name: postgresql
      state: started
```

Its syntax is easy and packages are pretty small. Relatively faster than Chef and Puppet.

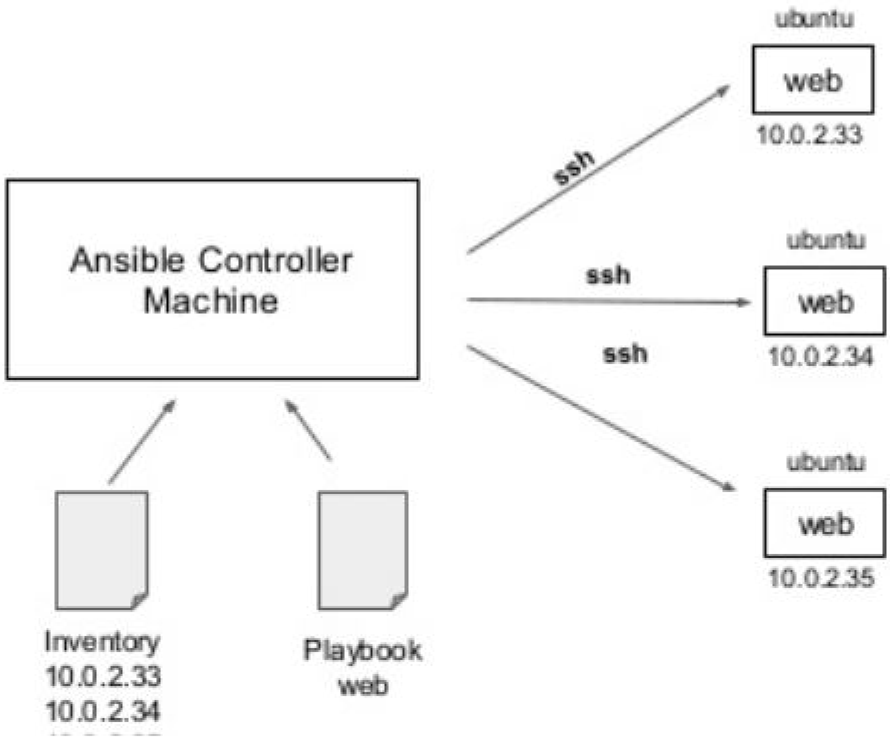


Figure 8 : Ansible’s Basic Structure

Chef

Chef works as a *settings 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 config management helper 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 reneges to a mutable infrastructure design and making it hard to examine configuration problems, but packer checks every update as a deployment of a new server configuration.

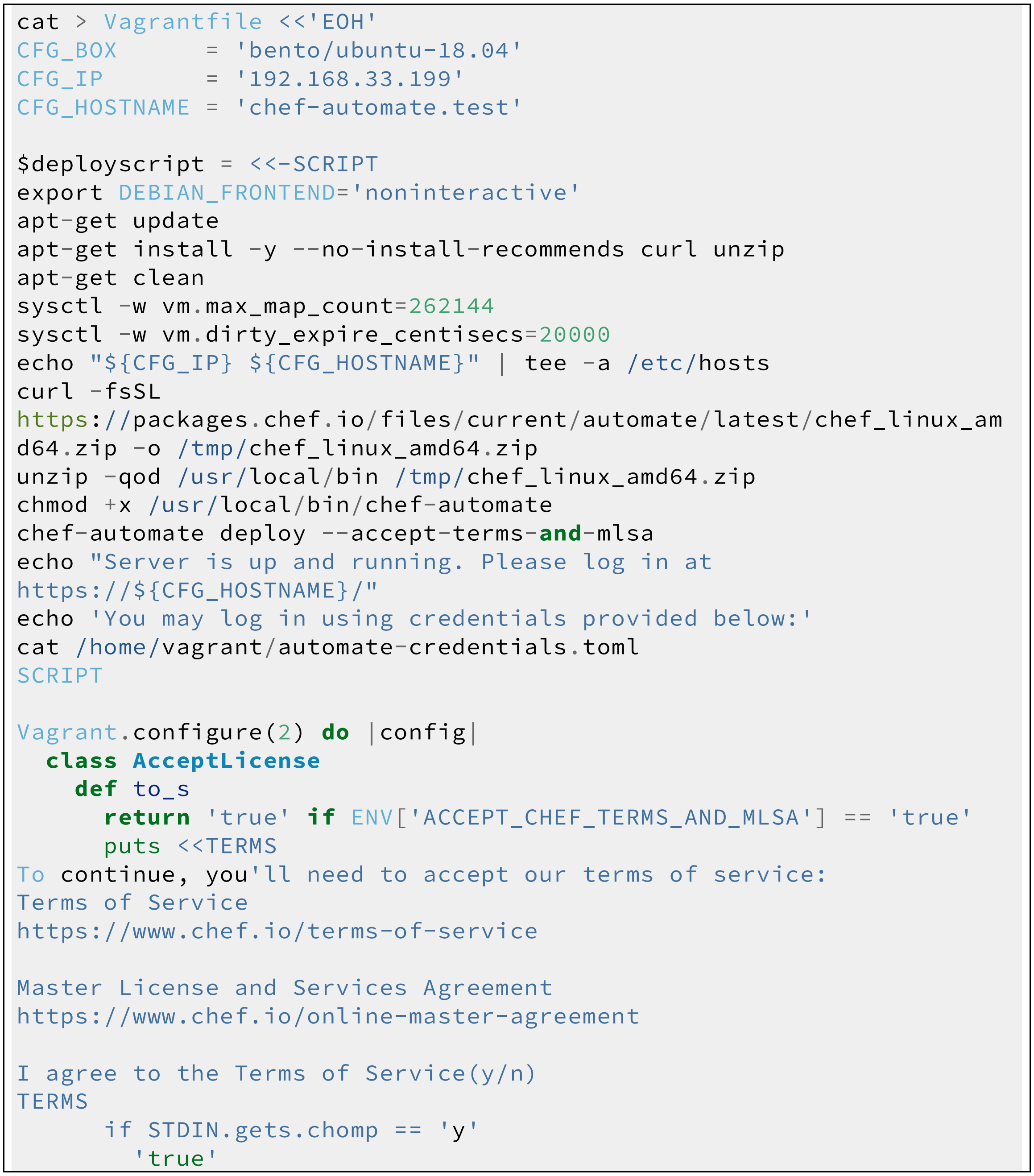
Chef uses an imperative language.

While packer don't have always to be a master role, Chef instantiate a procedural language and needs to be as master server for running state storing also for the agent application on every server.

Chef has three crucial components; Masters, The Chef Server and the Worker Nodes.

Chef works on vagrantfiles and can be accessible with Vagrantbox's.

Here is a basic implementation of Chef as vagrantfile :



```
cat > Vagrantfile <<'EOH'
CFG_BOX      = 'bento/ubuntu-18.04'
CFG_IP       = '192.168.33.199'
CFG_HOSTNAME = 'chef-automate.test'

$deployscript = <<-SCRIPT
export DEBIAN_FRONTEND='noninteractive'
apt-get update
apt-get install -y --no-install-recommends curl unzip
apt-get clean
sysctl -w vm.max_map_count=262144
sysctl -w vm.dirty_expire_centisecs=20000
echo "${CFG_IP} ${CFG_HOSTNAME}" | tee -a /etc/hosts
curl -fsSL
https://packages.chef.io/files/current/automate/latest/chef_linux_am
d64.zip -o /tmp/chef_linux_amd64.zip
unzip -qod /usr/local/bin /tmp/chef_linux_amd64.zip
chmod +x /usr/local/bin/chef-automate
chef-automate deploy --accept-terms-and-mlsa
echo "Server is up and running. Please log in at
https://${CFG_HOSTNAME}/"
echo 'You may log in using credentials provided below:'
cat /home/vagrant/automate-credentials.toml
SCRIPT

Vagrant.configure(2) do |config|
  class AcceptLicense
    def to_s
      return 'true' if ENV['ACCEPT_CHEF_TERMS_AND_MLSA'] == 'true'
      puts <<TERMS
To continue, you'll need to accept our terms of service:
Terms of Service
https://www.chef.io/terms-of-service

Master License and Services Agreement
https://www.chef.io/online-master-agreement

I agree to the Terms of Service(y/n)
TERMS
      if STDIN.gets.chomp == 'y'
        'true'
      else
        puts 'Chef Software Terms of Service and Master License and
Services Agreement were not accepted'
        exit 1
      end
    end
  end
  config.vm.box      = CFG_BOX
  config.vm.hostname = CFG_HOSTNAME

  config.vm.provider "virtualbox" do |v|
    v.name       = 'chef-automate'
    v.memory     = 4096
    v.cpus       = 2
    v.customize ['modifyvm', :id, '--audio', 'none']
  end

  config.vm.synced_folder '.', '/opt/a2-testing', create: true
  config.vm.network       'private_network', ip: CFG_IP
  config.vm.provision     'shell', env: {'CFG_IP' => CFG_IP,
                                         'CFG_HOSTNAME' =>
CFG_HOSTNAME,

'ACCEPT_CHEF_TERMS_AND_MLSA' => AcceptLicense.new}, inline:
$deployscript
end
EOH
```

Puppet

Puppet is a tool accesible in both open source and as paid version and it has its own language as interpreter.
However to use in extended versions, we must pay so it has a disadvantage to Packer, since Packer is completely open source.
Unlike chef, Puppet adapts a declarative,model based language that is similar to JSON.
You describe the resource’s state but you cannot interact in how this state is actually done
With Puppet, we create *manifests* and modules, while in Chef we create *recipes* and *cookbooks*.
Puppet is a configuration management tool too and its syntax is differentiated from the underlying OS and its applications' syntax. This leads you to define high-level concepts such as user, application, and service.
Puppet adapts every needs as data. Node's current state and the node’s targeted end-state, and the steps needed to step from one-by-one.
However since Puppet is not very useful on immutable systems, we must eliminate this as an option too.

2.1.2.Immutable

Terraform

As Packer, Terraform also produced by Hashicorp as open source provisioning tool. Terraform is an open-source IaSC tool that provides a consistent CLI automation workflow to manage multiple cloud services altogether. Since it convey IaC mentality, Terraform codifies cloud APIs into declarative configuration parts and files. It can work with Packer or alone.

In contrast with Packer, Terraform uses HashiCorp Configuration Language (HCL)

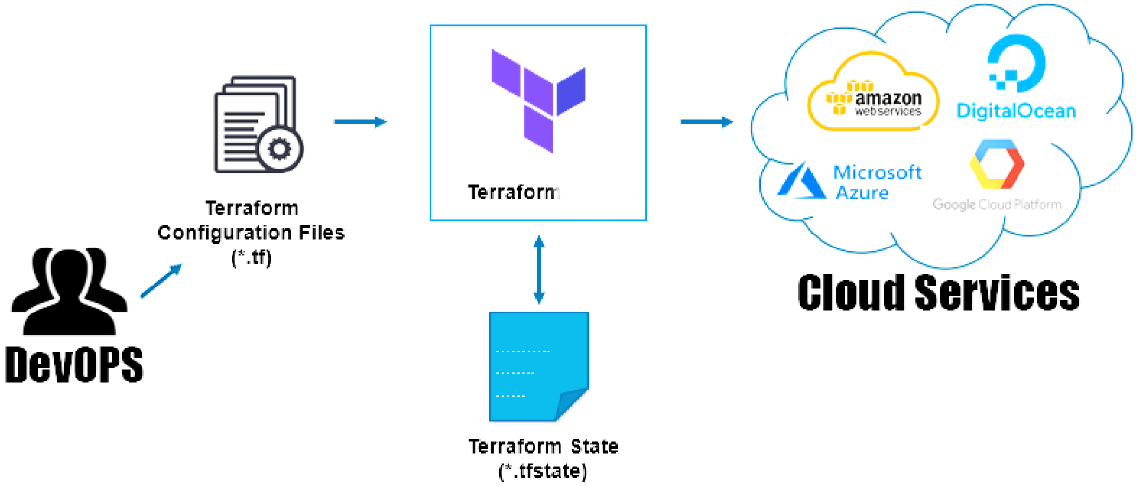


Figure 9 : Basic Structure of Terraform

Terraform is better than Ansible,Puppet and Chef when it comes to configuration because it also has a feature to manage orchestration.[5]

Terraform uses its own file format while Packer can easily worked with only JSON file.

Basic code structure of a Terraform unit ;

```
└──── tf/
   ├──── modules/
   │     ├──── network/
   │     │    ├──── main.tf
   │     │    ├──── dns.tf
   │     │    ├──── outputs.tf
   │     │    └──── variables.tf
   │     └──── spaces/
   │          ├──── main.tf
   │          ├──── outputs.tf
   │          └──── variables.tf
   └──── applications/
      ├──── backend-app/
      │    ├──── env/
      │    │    ├──── dev.tfvars
      │    │    ├──── staging.tfvars
      │    │    ├──── qa.tfvars
      │    │    └──── production.tfvars
      │    └──── main.tf
      └──── frontend-app/
         ├──── env/
         │    ├──── dev.tfvars
         │    ├──── staging.tfvars
         │    ├──── qa.tfvars
         │    └──── production.tfvars
```

```
            └──── main.tf
```

2.2. Overall Problems of Existing Systems

While HashiCorp Packer automates the creation of *any* kind of VMs and their configuration, other tools not yet covering all systemic settings. Packer embraces modern configuration management, orchestration in the understanding of Immutable Infrastructure&Mutable Infrustructure by encouraging DevOps teams to use automated scripts as pre-baked templates and encouraging to deploy and configure the applications within your pre-baked (Packer-made) VMIs on any platform.

Packer brings VMIs into the contemporary age, release the potentials of cloud systems opening new opportunities like traffic enhancing.

Packer only use a simple JSON file to achieve all its promises while Terraform uses its own format.

While others don’t have these abilities, Packer does all with its growing communities as an open source tool.

Provisioning, configuring is the easiest since its learning curve is amongst the best.[9]

Packer is free, small, platform independent and no need to be master kernel needs to manipulate a deployment on any SaaC.

2.3. Comparison Between Existing and Proposed Method

Table 2.1: Comparison of methods

Packer	Terraform	Ansible	Puppet	Chef
Can Work Alone	Can Work Alone	Cannot	Cannot	Cannot
Cross platform builds	Cross platform builds	Cross platform builds	Cross platform builds	Cross platform builds
Provisioning & Orchestration	Provisioning & Orchestration	Config Mngmnt	Config Mngmnt	Config Mngmnt

Table 2.2: Comparison of methods

	Packer	Terraform	Ansible	Puppet	Chef
Source	Open	Open	Open	Open	Open
Cloud	All	All	All	All	All
Infrastructure	Immutable	Immutable	Mutable	Mutable	Mutable
Language	Declarative	Declarative	Procedural	Declarative	Procedural
Agent	No	No	Yes	No	No
Master	No	No	Yes	No	No
Community	Large	Large	Large	Large	Large
Maturity	High	High	High	High	High

Note that, Table and Figure numbers start with the Section number and continue with the unique float identifier. All floats (Figures, Tables, Pseudocodes, Code Listings, etc.) should be referred in the text and if the float came from a different resource, you must provide a complete citation (e.g. Reprinted from [3]).

3. 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 built in 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.[23]

3.1. DESIGN OVERVIEW

Contemporary cloud services like AWS have built-in pre-baked VMs to serve customers desires.

Preloading required softwares,patches,settings and configurations on an Amazon EC2 VMI copy, then generating an image of that takes a couple of minute for each time.

After all, the resulting image might be used to create new copies (instances) with all software and configuration previously loaded. This process permits EC2 instance to be on-line and be ready for serve in a quick way. [21]

It's not only make it simple the deployment of new copies but also mainly handy when an instance is member of an Auto Scaling group and is it's reacting to a bug 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.

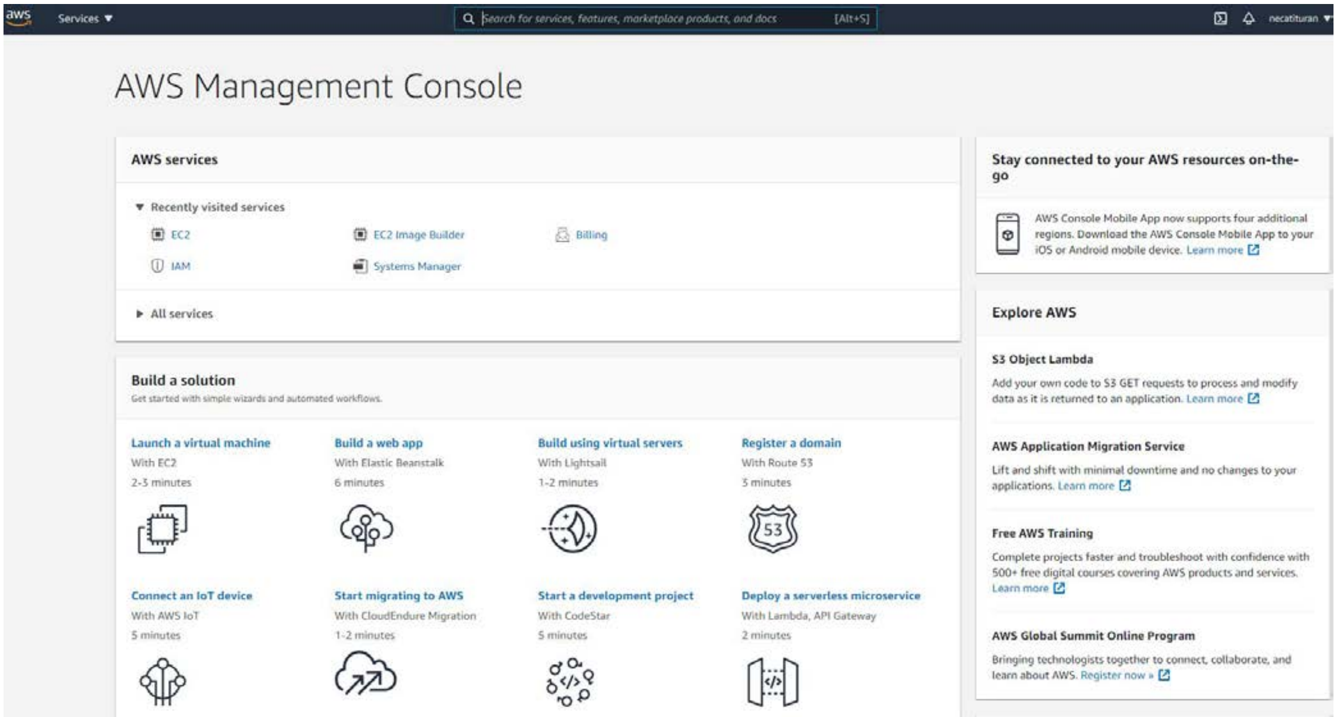
To this purpose we used Packer to pre-bake AMIs without interfering EC2 console.

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



AWS Console Mobile App now supports four additional regions. Download the AWS Console Mobile App to your iOS or Android mobile device. Learn more

Add your own code to S3 GET requests to process and modify data as it is returned to an application. Learn more

Lift and shift with minimal downtime and no changes to your applications. Learn more

Complete projects faster and troubleshoot with confidence with 500+ free digital courses covering AWS products and services. Learn more

Bringing technologists together to connect, collaborate, and learn about AWS. Register now »

Figure 10 : AWS Management CONSOLE

What is Amazon EC2?

Amazon Elastic Compute Cloud (Amazon EC2) give customers a expansible integrated computing service in the AWS Cloud ecosystem.

Using Amazon EC2 serve your obligations to fund in physical hardware supplies so we can develop and deploy applications faster without needing a physical equipment.

We can use AWS to create and launch as many as virtual systems as we need and we can configure entails with security, networking, storage management.

3.2. 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 settings of CPU, data storages, memories and networking volume for our copies, called as instance types
- Secure login confirmations for our copies using key pairs so we can secure our services
- Storage capacities for temp data (deleted as need to stop/hibernate/terminate our copies)
- 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 the purpose of dynamic cloud computing, called as Elastic IPs.
- Virtual networks (VPCs) we can make for our instances that are logically isolated from the rest of the cloud services.

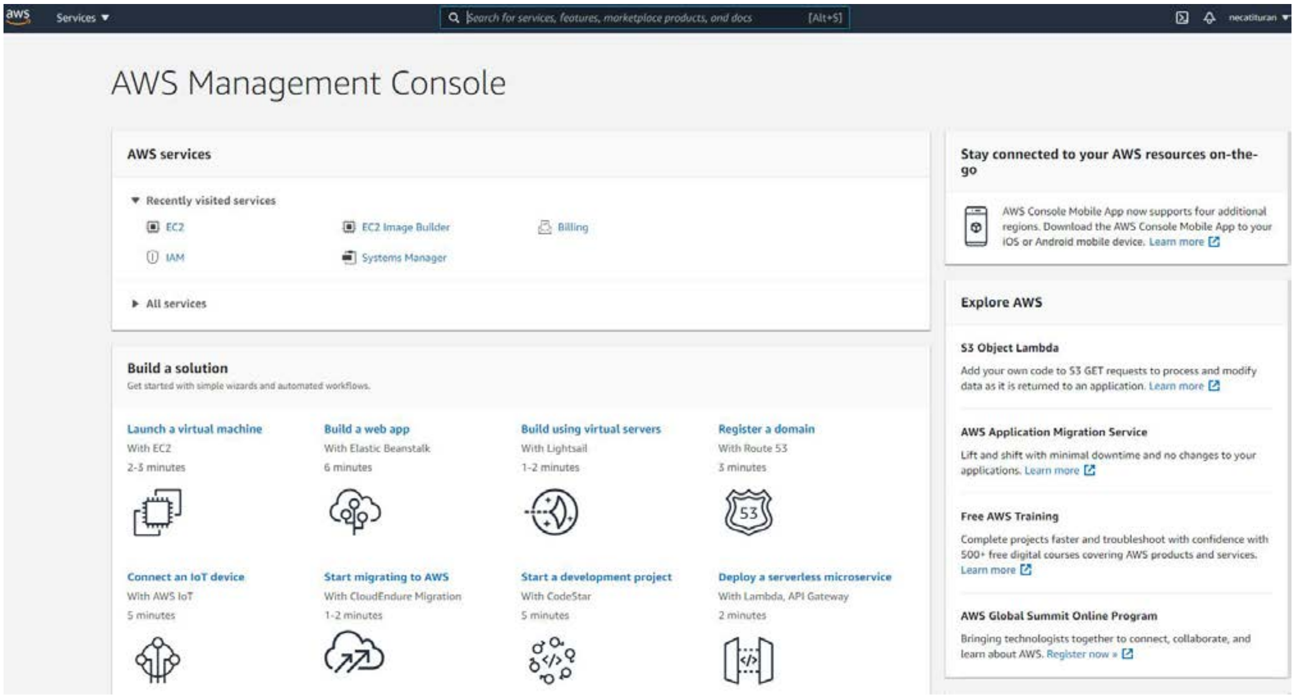
3.3. SYSTEM ARCHITECTURE

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 create an EC2 instance;

- Sign in to the Amazon and then go to the AWS Console.

- Go through EC2 console by choosing EC2 under Compute.


aws
Services ▼
Search for services, features, marketplace products, and docs [Alt+S]
necatituran ▼

AWS Management Console

AWS services

▼ Recently visited services

EC2 | EC2 Image Builder | Billing

IAM | Systems Manager

▶ All services

Build a solution

Get started with simple wizards and automated workflows.

Launch a virtual machine With EC2 2-3 minutes	Build a web app With Elastic Beanstalk 6 minutes	Build using virtual servers With Lightsail 1-2 minutes	Register a domain With Route 53 3 minutes
Connect an IoT device With AWS IoT 5 minutes	Start migrating to AWS With CloudEndure Migration 1-2 minutes	Start a development project With CodeStar 5 minutes	Deploy a serverless microservice With Lambda, API Gateway 2 minutes

Stay connected to your AWS resources on-the-go

AWS Console Mobile App now supports four additional regions. Download the AWS Console Mobile App to your iOS or Android mobile device. Learn more

Explore AWS

S3 Object Lambda

Add your own code to S3 GET requests to process and modify data as it is returned to an application. Learn more

AWS Application Migration Service

Lift and shift with minimal downtime and no changes to your applications. Learn more

Free AWS Training

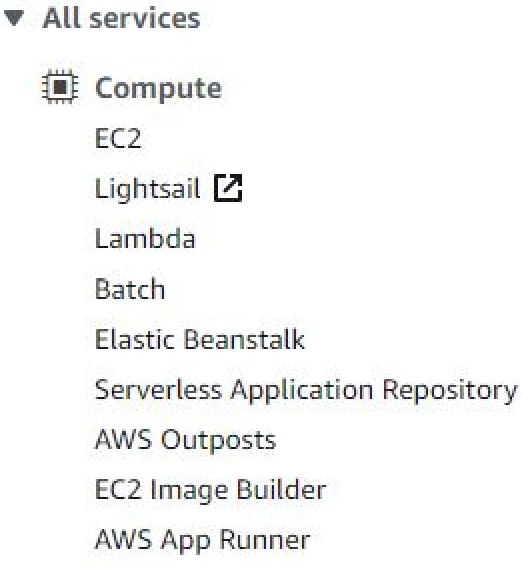
Complete projects faster and troubleshoot with confidence with 500+ free digital courses covering AWS products and services. Learn more

AWS Global Summit Online Program

Bringing technologists together to connect, collaborate, and learn about AWS. Register now »

Figure 11 : General view of AWS Management Console

Choose all services;



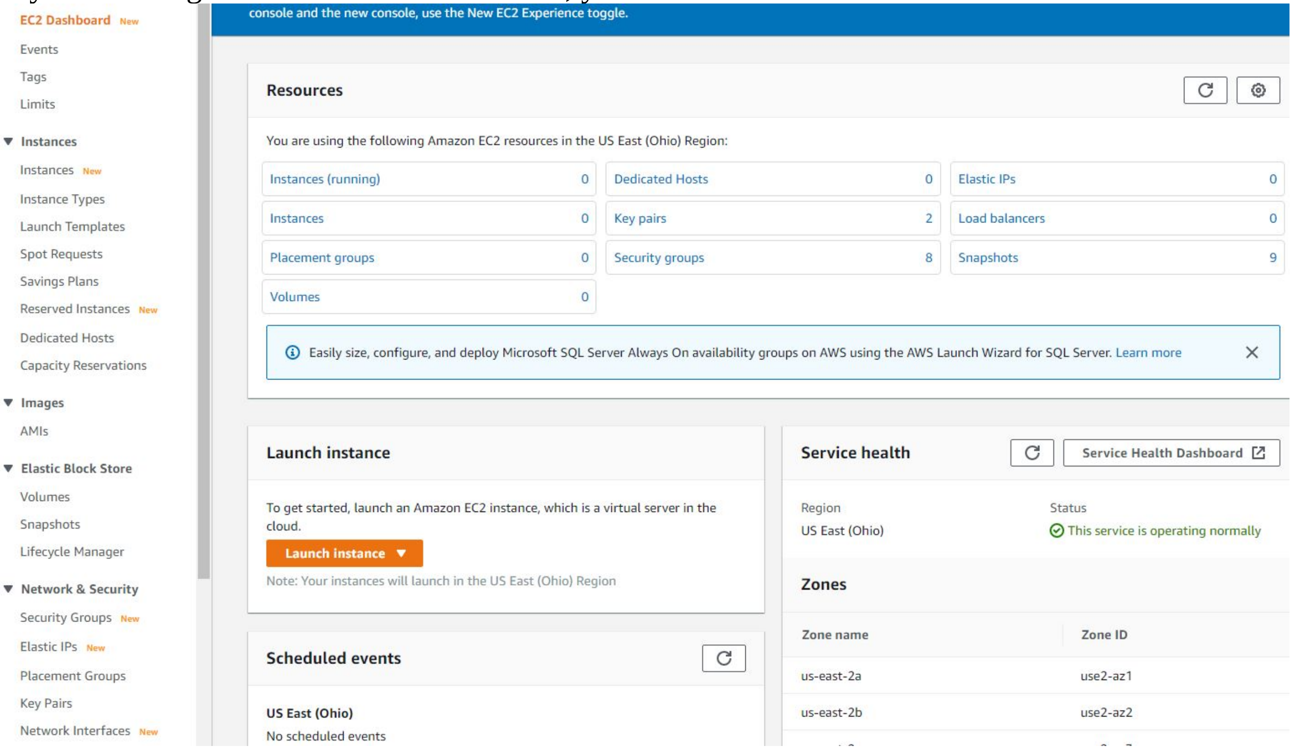
▼ All services

Compute

- EC2
- Lightsail
- Lambda
- Batch
- Elastic Beanstalk
- Serverless Application Repository
- AWS Outposts
- EC2 Image Builder
- AWS App Runner

Figure 12 : Services

If you are using the Show All Services view, your screen looks like this:



EC2 Dashboard New

Events

Tags

Limits

▼ Instances

- Instances New
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances New
- Dedicated Hosts
- Capacity Reservations

▼ Images

- AMIs

▼ Elastic Block Store

- Volumes
- Snapshots
- Lifecycle Manager

▼ Network & Security

- Security Groups New
- Elastic IPs New
- Placement Groups
- Key Pairs
- Network Interfaces New

console and the new console, use the New EC2 Experience toggle.

Resources

You are using the following Amazon EC2 resources in the US East (Ohio) Region:

Instances (running)	0	Dedicated Hosts	0	Elastic IPs	0
Instances	0	Key pairs	2	Load balancers	0
Placement groups	0	Security groups	8	Snapshots	9
Volumes	0				

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Launch instance ▼

Note: Your instances will launch in the US East (Ohio) Region

Scheduled events

US East (Ohio)

No scheduled events

Service health

Service Health Dashboard

Region: US East (Ohio)

Status: This service is operating normally

Zones

Zone name	Zone ID
us-east-2a	use2-az1
us-east-2b	use2-az2

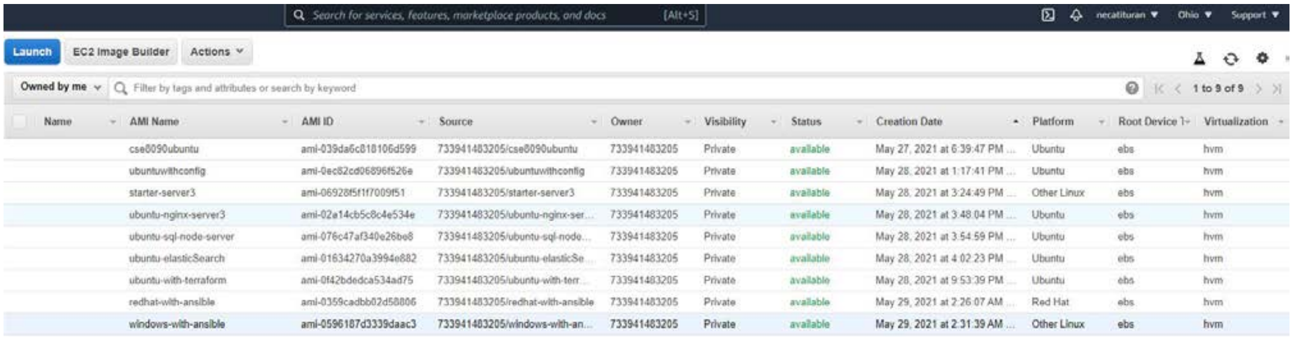
Figure 12 : EC2 Panel

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.



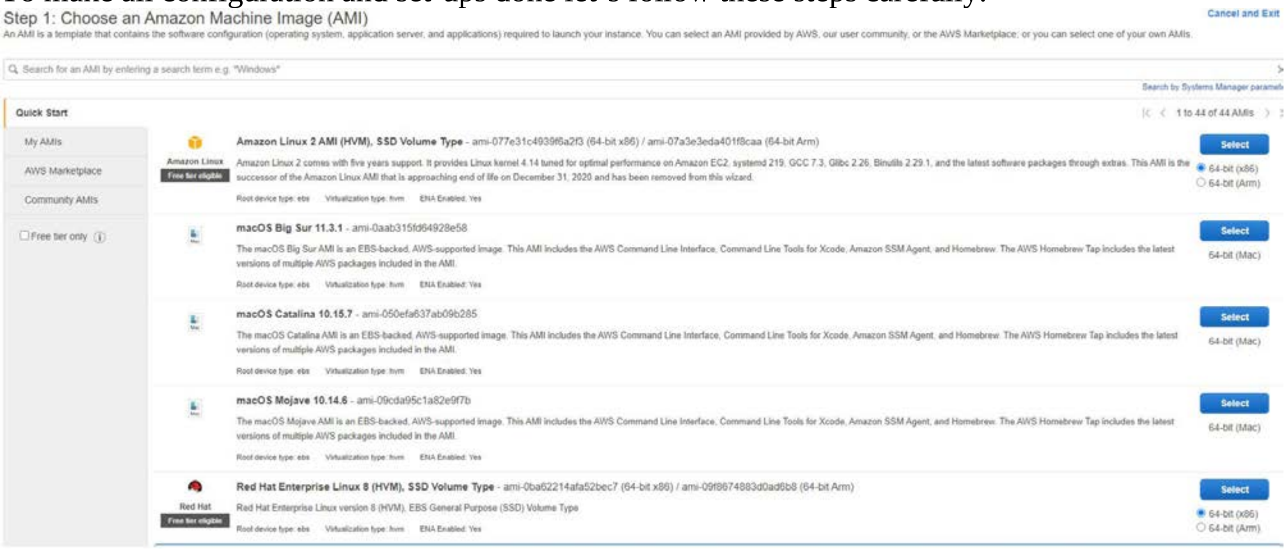
Search for services, features, marketplace products, and docs [Alt+S]
necatituran ▼
Ohio ▼
Support ▼
Launch
EC2 Image Builder
Actions
Owned by me
Filter by tags and attributes or search by keyword
1 to 9 of 9

Name	AMI Name	AMI ID	Source	Owner	Visibility	Status	Creation Date	Platform	Root Device T	Virtualization
	cse8090ubuntu	ami-039da6c818106d599	733941483205/cse8090ubuntu	733941483205	Private	available	May 27, 2021 at 6:39:47 PM ...	Ubuntu	ebs	hvm
	ubuntuwithconfig	ami-0ec82cd06896f526e	733941483205/ubuntuwithconfig	733941483205	Private	available	May 28, 2021 at 1:17:41 PM ...	Ubuntu	ebs	hvm
	starter-server3	ami-06928f5f1f7009f51	733941483205/starter-server3	733941483205	Private	available	May 28, 2021 at 3:24:49 PM ...	Other Linux	ebs	hvm
	ubuntu-nginx-server3	ami-02a14cb5c8c4e534e	733941483205/ubuntu-nginx-ser...	733941483205	Private	available	May 28, 2021 at 3:48:04 PM ...	Ubuntu	ebs	hvm
	ubuntu-sql-node-server	ami-076c47af340e26be8	733941483205/ubuntu-sql-node...	733941483205	Private	available	May 28, 2021 at 3:54:59 PM ...	Ubuntu	ebs	hvm
	ubuntu-elasticSearch	ami-01634270a3994e882	733941483205/ubuntu-elasticSe...	733941483205	Private	available	May 28, 2021 at 4:02:23 PM ...	Ubuntu	ebs	hvm
	ubuntu-with-terraform	ami-0f42bdedca534ad75	733941483205/ubuntu-with-terr...	733941483205	Private	available	May 28, 2021 at 9:53:39 PM ...	Ubuntu	ebs	hvm
	redhat-with-ansible	ami-0359cadbb02d58806	733941483205/redhat-with-ansible	733941483205	Private	available	May 29, 2021 at 2:26:07 AM ...	Red Hat	ebs	hvm
	windows-with-ansible	ami-0596187d3339daac3	733941483205/windows-with-an...	733941483205	Private	available	May 29, 2021 at 2:31:39 AM ...	Other Linux	ebs	hvm

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.



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-077e31c4939f6a2f3 (64-bit x86) / ami-07a3e3eda401f8caa (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

● 64-bit (x86)
○ 64-bit (Arm)

macOS Big Sur 11.3.1 - ami-0aab315fd64928e58

The macOS Big Sur AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

64-bit (Mac)

macOS Catalina 10.15.7 - ami-050efa637ab09b285

The macOS Catalina AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

64-bit (Mac)

macOS Mojave 10.14.6 - ami-09cda95c1a82e9f7b

The macOS Mojave AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

64-bit (Mac)

Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0ba62214afa52bec7 (64-bit x86) / ami-09f8674883d0ad6b8 (64-bit Arm)

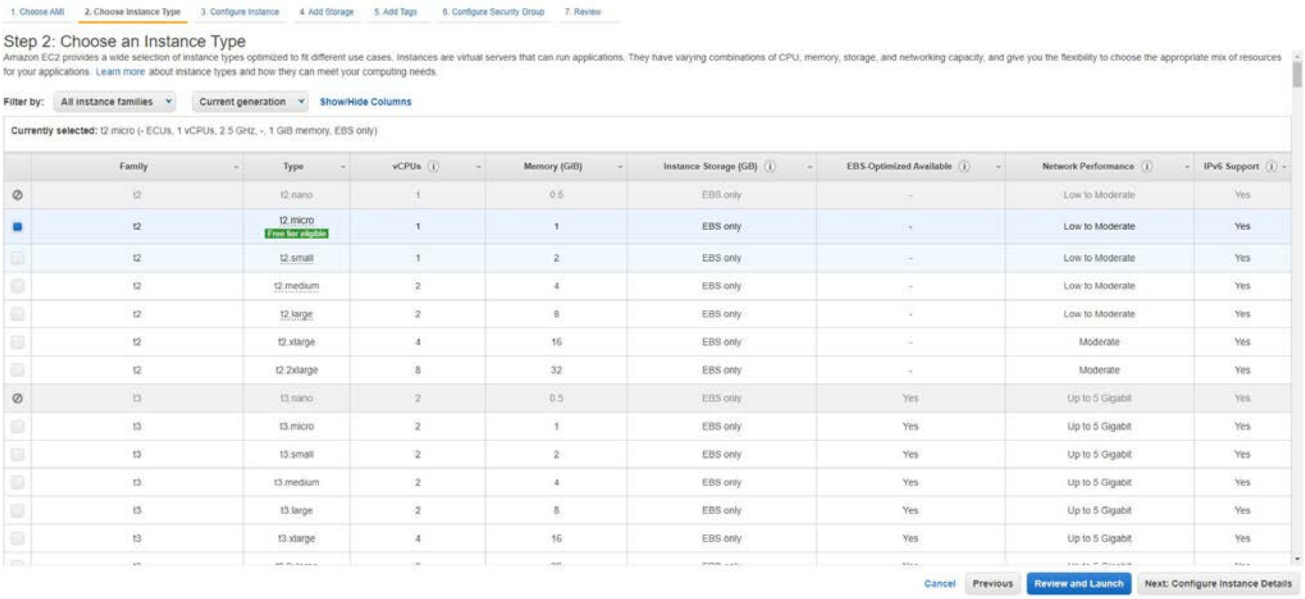
Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

● 64-bit (x86)
○ 64-bit (Arm)

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.



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

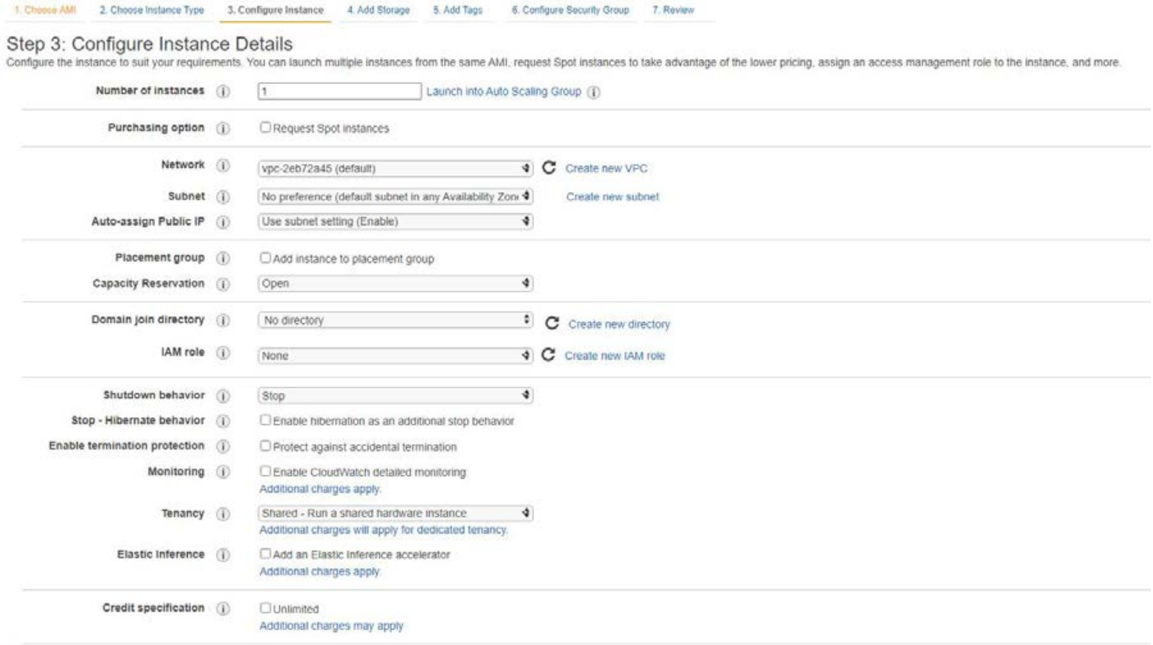
Filter by: All instance families ▾ Current generation ▾ Show/Hide Columns

Currently selected: t2 micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
⊘	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
■	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
□	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
□	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
□	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
□	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
□	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
⊘	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.medium	2	4	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.large	2	8	EBS only	Yes	Up to 5 Gigabit	Yes
□	t3	t3.xlarge	4	16	EBS only	Yes	Up to 5 Gigabit	Yes

Figure 15 : AMI Type

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



Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances: 1 — Launch into Auto Scaling Group

Purchasing option: □ Request Spot instances

Network: vpc-2eb72a45 (default) — Create new VPC

Subnet: No preference (default subnet in any Availability Zon — Create new subnet

Auto-assign Public IP: Use subnet setting (Enable)

Placement group: □ Add instance to placement group

Capacity Reservation: Open

Domain join directory: No directory — Create new directory

IAM role: None — Create new IAM role

Shutdown behavior: Stop

Stop - Hibernate behavior: □ Enable hibernation as an additional stop behavior

Enable termination protection: □ Protect against accidental termination

Monitoring: □ Enable CloudWatch detailed monitoring
Additional charges apply.

Tenancy: Shared - Run a shared hardware instance
Additional charges will apply for dedicated tenancy.

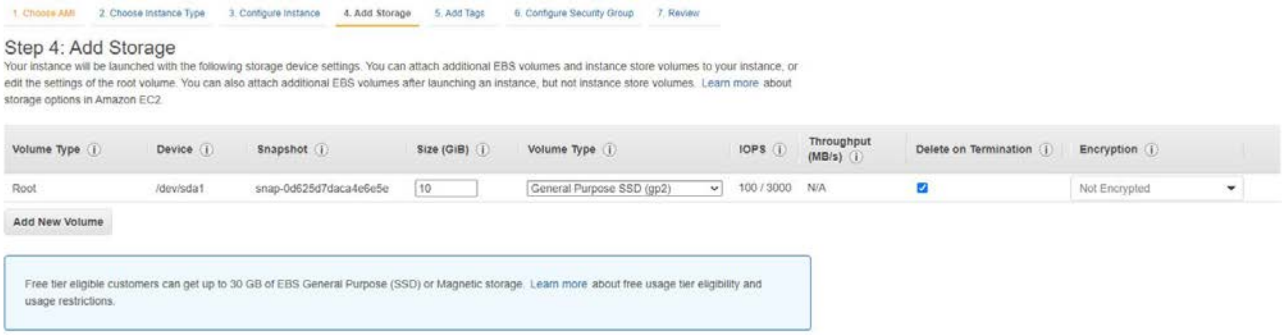
Elastic Inference: □ Add an Elastic Inference accelerator
Additional charges apply.

Credit specification: □ Unlimited
Additional charges may apply

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.[9]

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.[17]



Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

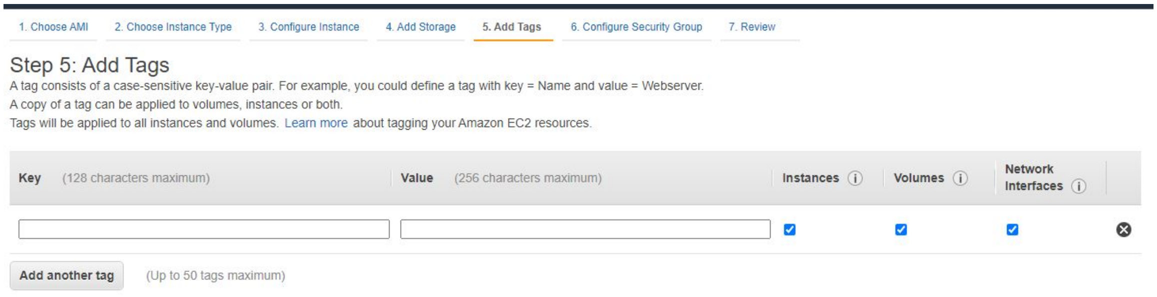
Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-0d625d7daca4e6e5e	10	General Purpose SSD (gp2)	100 / 3000	N/A	☑	Not Encrypted

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

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.



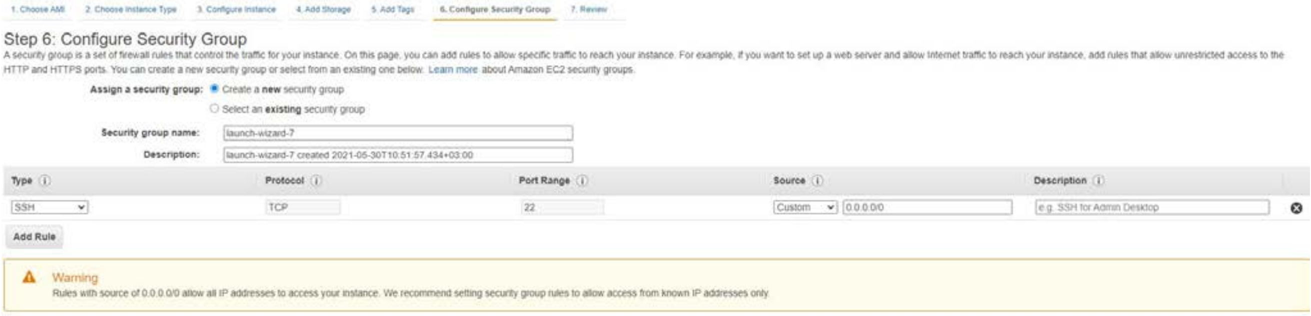
Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
A copy of a tag can be applied to volumes, instances or both.
Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances	Volumes	Network Interfaces
		☑	☑	☑

(Up to 50 tags maximum)

Figure 18 : Tags of AMI



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: ● Create a **new** security group
○ Select an **existing** security group

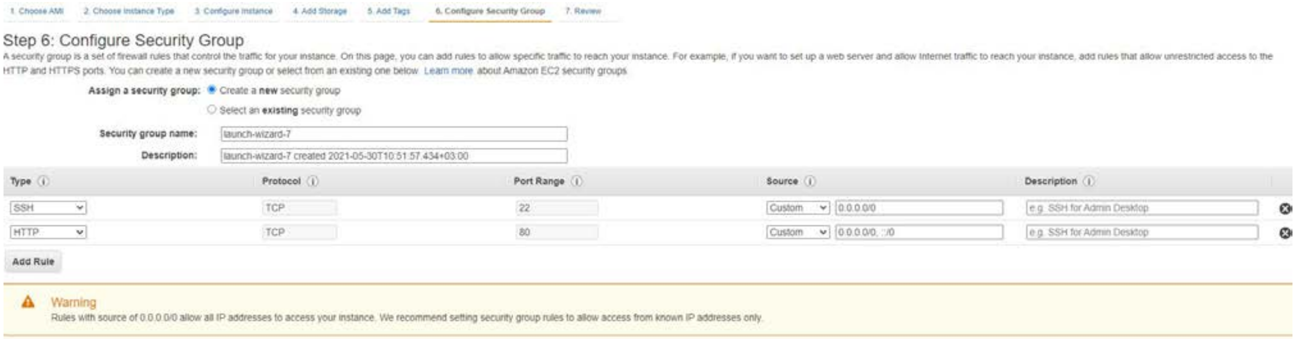
Security group name: launch-wizard-7
Description: launch-wizard-7 created 2021-05-30T10:51:57.434+03:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

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.[6][4]



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

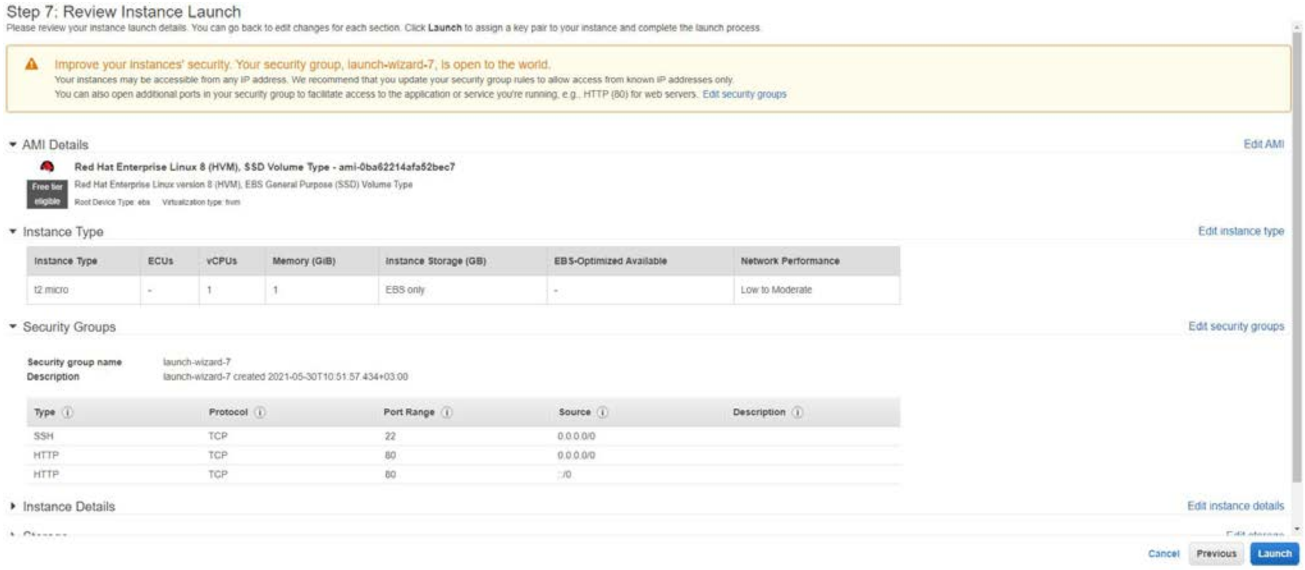
Assign a security group: ● Create a **new** security group
○ Select an **existing** security group

Security group name: launch-wizard-7
Description: launch-wizard-7 created 2021-05-30T10:51:57.434+03:00

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Figure 20 : Adding HTTP Rule



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Your security group, launch-wizard-7, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0ba62214afa52bec7
Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type
Root Device Type: ebs Virtualization type: hvm

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

Security group name launch-wizard-7
Description launch-wizard-7 created 2021-05-30T10:51:57.434+03:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	::/0	

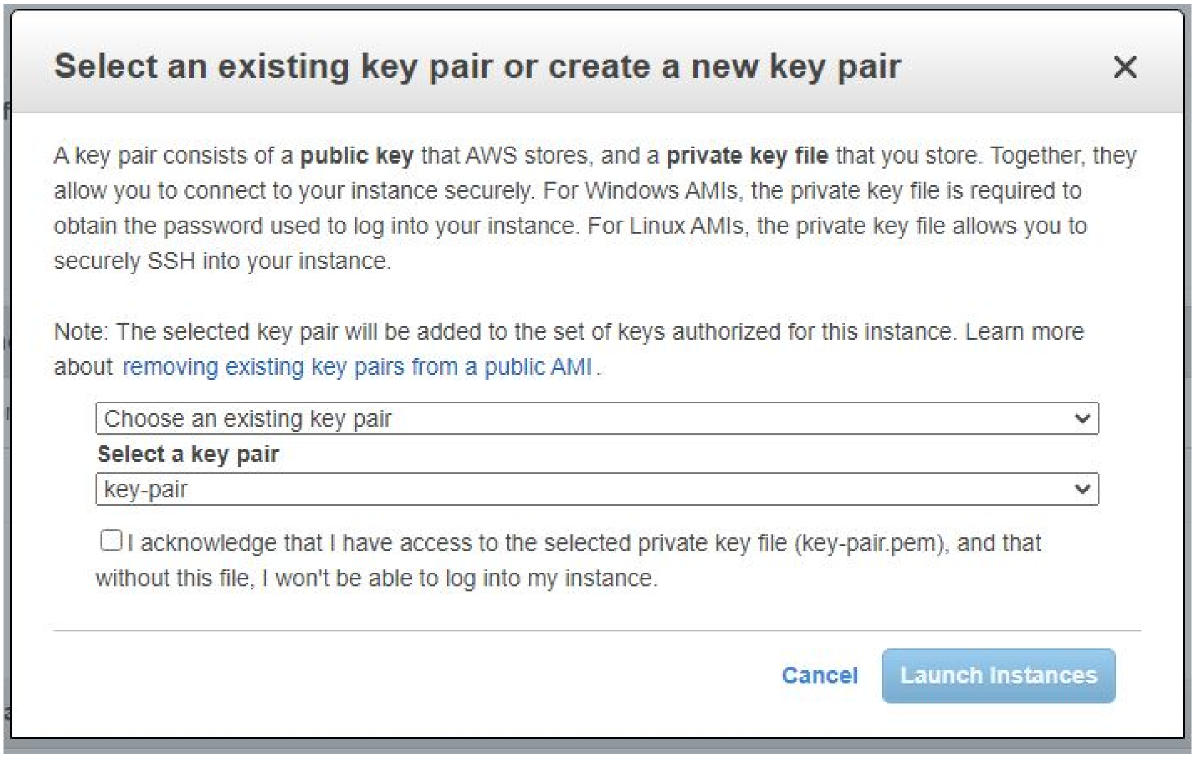
Figure 21 : Adding HTTPS Rule

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-----
(Primary SSL certificate: your_domain_name.crt)
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
(Your Intermediate certificate: DigiCertCA.crt)
-----END CERTIFICATE-----
```

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



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI .

Choose an existing key pair

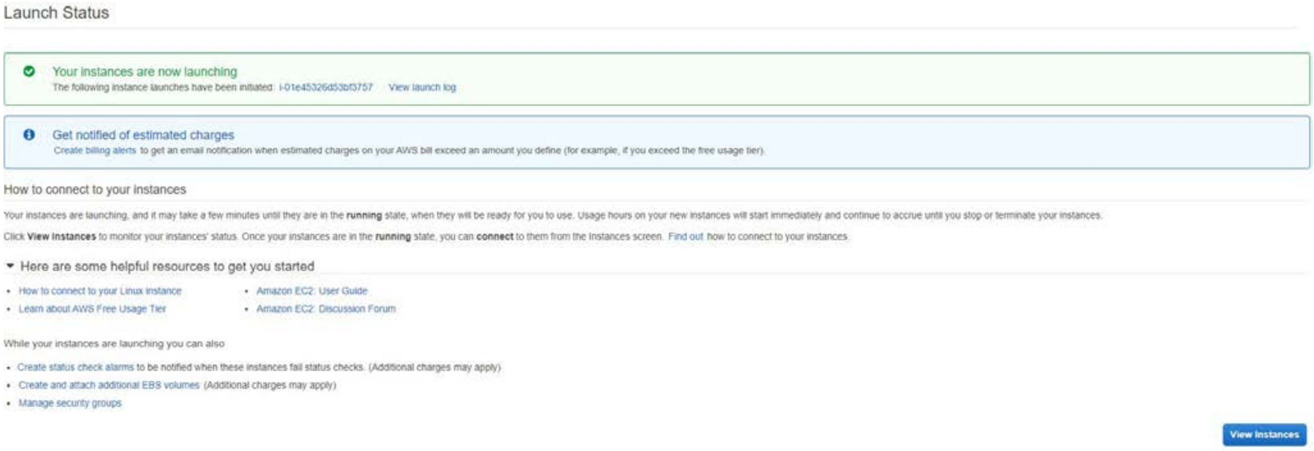
Select a key pair

key-pair

□ I acknowledge that I have access to the selected private key file (key-pair.pem), and that without this file, I won't be able to log into my instance.

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.



Launch Status

Your instances are now launching
The following instance launches have been initiated: i-01e45326d53bf3757 View launch log

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. Find out how to connect to your instances.

Here are some helpful resources to get you started

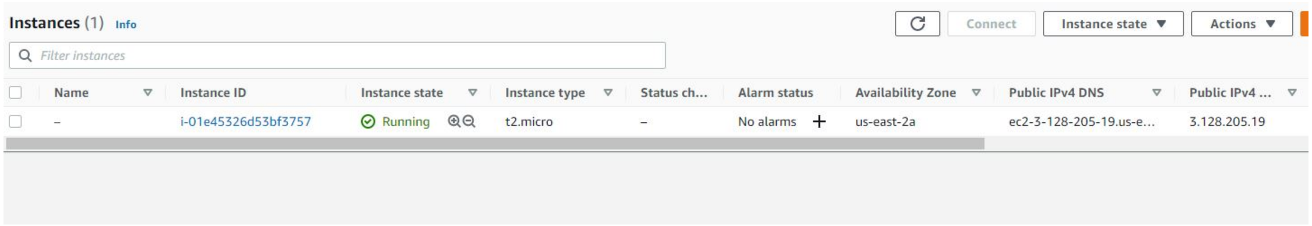
- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2: User Guide
- Amazon EC2: Discussion Forum

While your instances are launching you can also

- Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)
- Create and attach additional EBS volumes (Additional charges may apply)
- Manage security groups

Figure 23 : End of the creation of AMI

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



Instances (1) Info

□	Name	Instance ID	Instance state	Instance type	Status ch...	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
□	–	i-01e45326d53bf3757	Running	t2.micro	–	No alarms	us-east-2a	ec2-3-128-205-19.us-e...	3.128.205.19

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.

3.4. MODULE B

USING PACKER

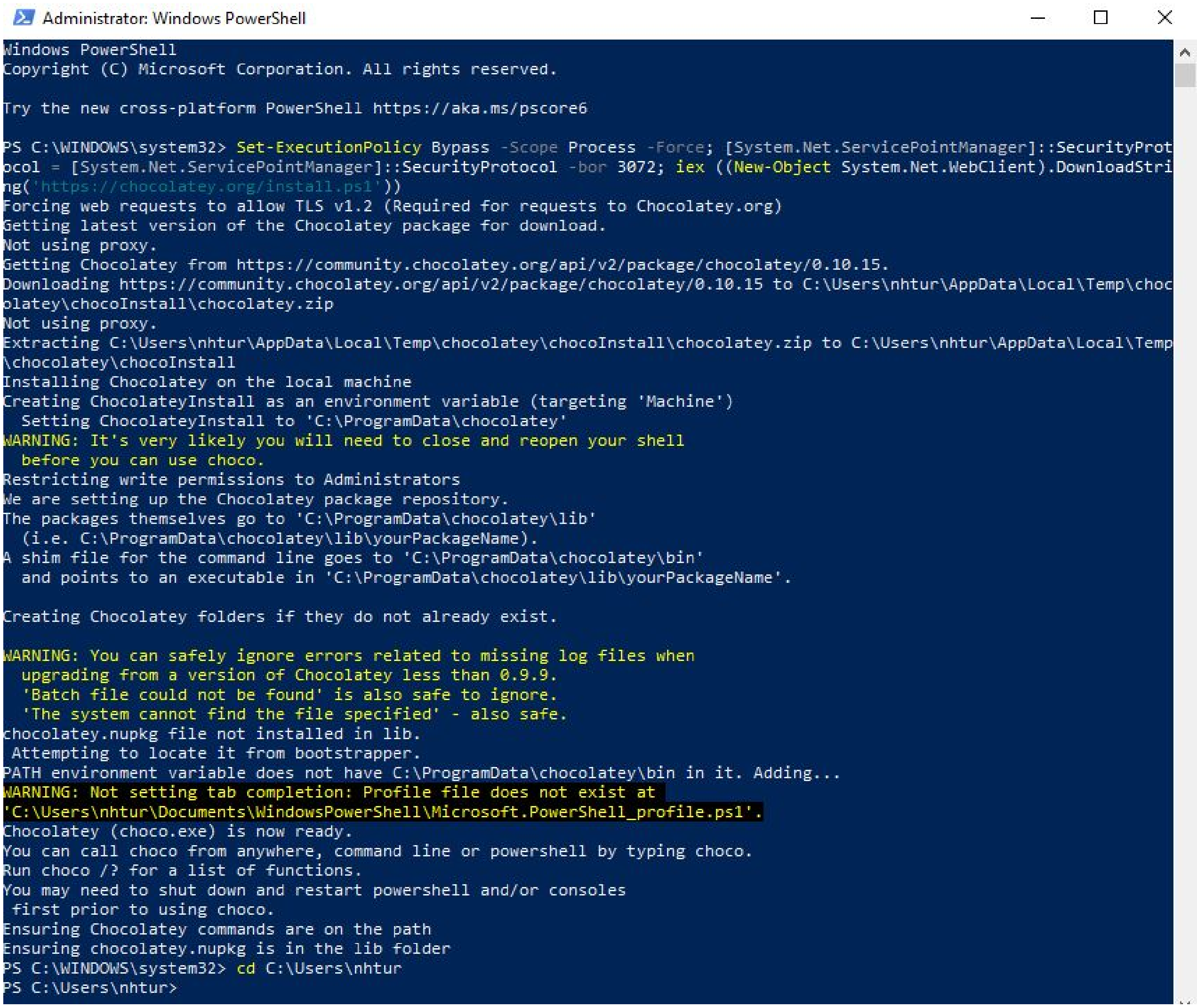
Packer can be used in the numerous ways:

- 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'))
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProt
ocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadStri
ng('https://chocolatey.org/install.ps1'))
Forcing web requests to allow TLS v1.2 (Required for requests to Chocolatey.org)
Getting latest version of the Chocolatey package for download.
Not using proxy.
Getting Chocolatey from https://community.chocolatey.org/api/v2/package/chocolatey/0.10.15.
Downloading https://community.chocolatey.org/api/v2/package/chocolatey/0.10.15 to C:\Users\nhtur\AppData\Local\Temp\choc
olatey\chocoInstall\chocolatey.zip
Not using proxy.
Extracting C:\Users\nhtur\AppData\Local\Temp\chocolatey\chocoInstall\chocolatey.zip to C:\Users\nhtur\AppData\Local\Temp
\chocolatey\chocoInstall
Installing Chocolatey on the local machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
  Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
  before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
  (i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
  and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.

Creating Chocolatey folders if they do not already exist.

WARNING: You can safely ignore errors related to missing log files when
  upgrading from a version of Chocolatey less than 0.9.9.
  'Batch file could not be found' is also safe to ignore.
  'The system cannot find the file specified' - also safe.
chocolatey.nupkg file not installed in lib.
 Attempting to locate it from bootstrapper.
PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding...
WARNING: Not setting tab completion: Profile file does not exist at
'C:\Users\nhtur\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'.
Chocolatey (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart powershell and/or consoles
 first prior to using choco.
Ensuring Chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
PS C:\WINDOWS\system32> cd C:\Users\nhtur
PS C:\Users\nhtur>
```

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

1.1.1. Continuous Delivery

Packer is small, transferable, and command-line driven so it’s easy to use remotely from any platform.

This makes Packer the best tool to put in the CD pipeline.

Packer also might be used to create 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.

So we can first test and deploy and if requirements are done and tests passed, then we can be sure about the the image will work when it’s deployed on the server. This conducts a new kind of stability,testability and configurability to infrastructure mutations.

1.1.2. 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 simultaneously.

So if we use Amazon Web Services for production but some other VMware for dev, we can generate both an AMI and a VMware image using Packer simultaneously from same Packer template with a JSON file.

1.1.3. Appliance/Demo Creation

Since the tool creates consistent VMIs for more than once platform in parallel, it’s the best tool for creating applications as one-use product demos for our services.

As software changes, DevOPS teams can automatically generate applications with the software pre-baked (installed).

This means that users can get ready with new software by deploying it to the cloud service of their any desires.

Packer’s Structure

Packer has 6 fundamental component in its hierarchy;

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

Builders

Builders are used for generating VMIs and creating images from different platforms.

e.g. EC2, VMware, VirtualBox.

Packer has many builders by default but DevOPS teams might also be extended to add new builders for their own purposes.

With packer-init plugin;

```
packer {
  required_plugins {
    amazon = {
      version = ">= 0.0.1"
      source = "github.com/hashicorp/amazonAWS"
    }
  }
}
```

On the command-line for simple AWS builder;

```
"builders": {
  "type": "amazon-ebs",
  "access_key": "A+53dKIAIOSdw2FODNN7EXAMPLE",
  "secret_key": "MDwJaEMMPLIK7MDENG/bMPxRfiCYEXAMPLEKEY",
  "region": "us-east-2",
}
```

Provisioners

Provisioners use built-in and 3rd party applications to install and configurate the VMI after the installation and booting it.

Basically provisioners make the system ready for any usage purpose, so common scenarios for provisioners are these:

- managing plug-ins
- changing the OS kernel
- attending and generating new users
- installing&downloading application codes

A simple shell provisioner example ;

```
{
  "type": "shell",
  "inline": ["echo foo"]
}
```

Templates – Variables

User variables make sure DevOPS teams can template their system to be further set with custom-defined variables from the coding interpreters like PowerShell,BASH etc., environment variables, Vaults, or directories as files.

This allows DevOPS teams create a framework from custom templates in order to save save tokens, environments and platform specific objects also any other types of information out of custom specific templates.

This feature maximizes the movable of the template for our needs.

```
{
  "variables": {
    "aws_access_key": "",
    "aws_secret_key": ""
  },

  "builders": [
    {
      "type": "amazon-ebs",
      "access_key": "{{user `aws_access_key`}}",
      "secret_key": "{{user `aws_secret_key`}}"
      // ...
    }
  ]
}
```

Vault tokens can be used variables too;

```
"VAULT_ADDR"
"VAULT_AGENT_ADDR"
"VAULT_CACERT"
"VAULT_CAPATH"
"VAULT_CLIENT_CERT"
"VAULT_CLIENT_KEY"
"VAULT_CLIENT_TIMEOUT"
"VAULT_SKIP_VERIFY"
"VAULT_NAMESPACE"
"VAULT_TLS_SERVER_NAME"
"VAULT_WRAP_TTL"
"VAULT_MAX_RETRIES"
"VAULT_TOKEN"
"VAULT_MFA"
```

```
"VAULT_RATE_LIMIT"
```

Paralel Builds

Parellel build is a very powerful and crucial feature of Packer and it makes Packer better than others.

Packer can build an AMI and a Azure Image virtual machine in parallel provisioned with the same templates (JSON scripts) and this make them near-identical images.

Imagine requiring using 2 different platform but we can remotely create them with the same scripts as a JSON file.

```
source "docker" "ubuntu-bionic" {
  image  = "ubuntu:bionic"
  commit = true
}

build {
  sources = [
-    "source.docker.ubuntu"
+    "source.docker.ubuntu",
+    "source.docker.ubuntu-bionic",
  ]
  ## ...
}
```

Post-Processors

It’s similar to the provisioners. Post-processors run after the VMI is built by the packer builder and by the provisioner.

Post-processors are not a necessity but it shows a powerful option. They might be used to upload artifacts, packages, or any other needs.

```
build {
  #build image
  post-processor "checksum" {
    checksum_types = [ "md5", "sha512" ]
    keep_input_artifact = true
 }

  post-processor "amazon-import" { # upload image to amazon
  }
}
```

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.

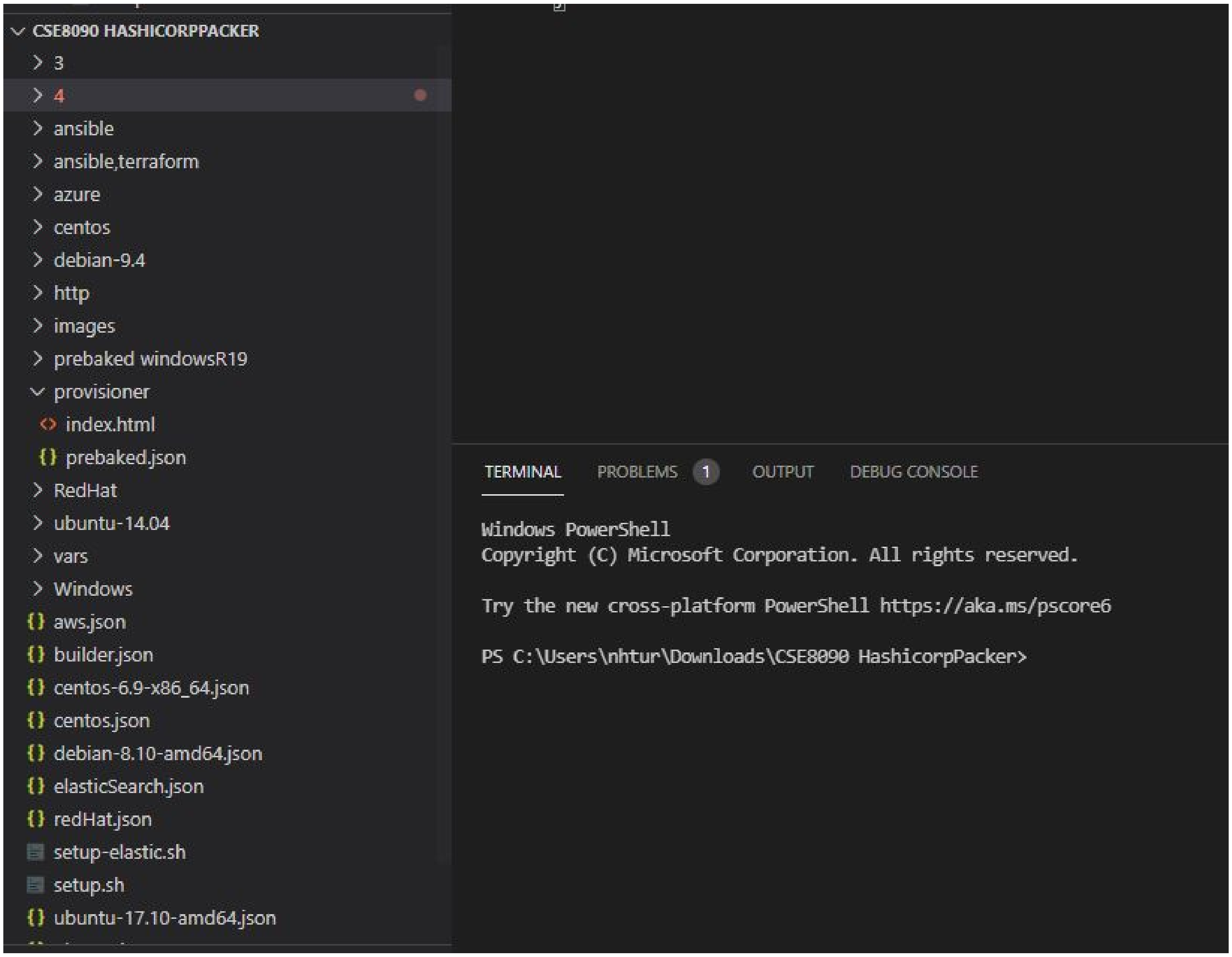
System Tools & Specifications

- 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.

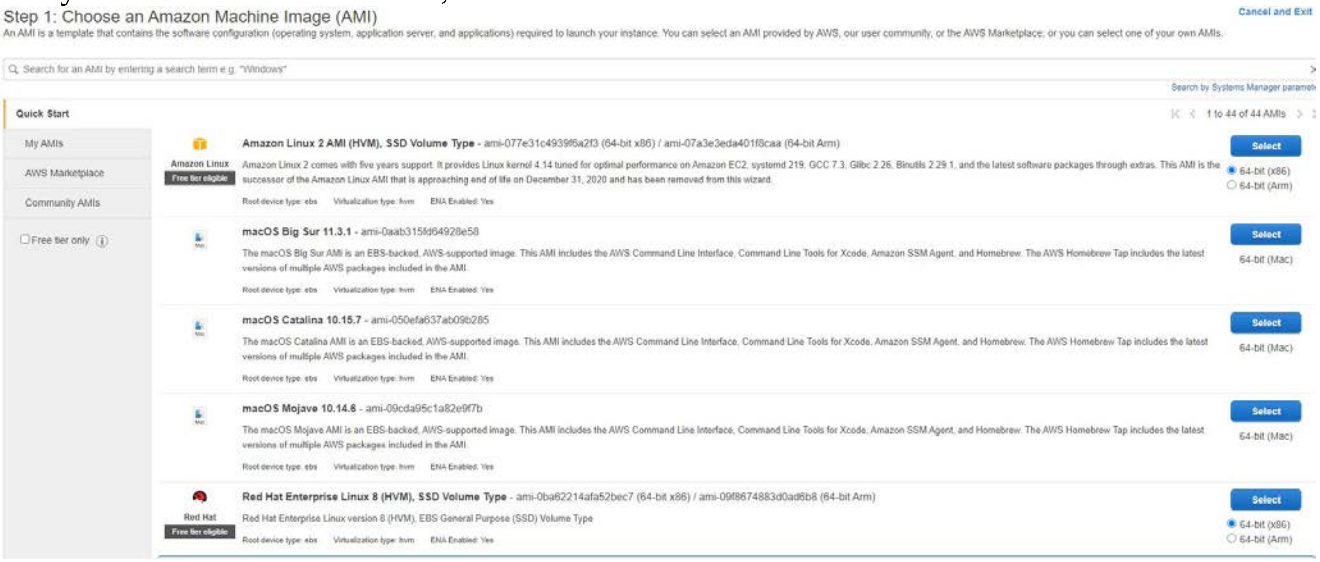


```
CSE8090 HASHICORPPACKER
  > 3
  > 4
  > ansible
  > ansible,terraform
  > azure
  > centos
  > debian-9.4
  > http
  > images
  > prebaked windowsR19
  v provisioner
     <> index.html
     {} prebaked.json
  > RedHat
  > ubuntu-14.04
  > vars
  > Windows
  {} aws.json
  {} builder.json
  {} centos-6.9-x86_64.json
  {} centos.json
  {} debian-8.10-amd64.json
  {} elasticSearch.json
  {} redHat.json
  setup-elastic.sh
  setup.sh
  {} ubuntu-17.10-amd64.json
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\nhtur\Downloads\CSE8090 HashicorpPacker>
```

Figure 25 : Setting The VS Code for Packer

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



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-077e31c4939f6a2f3 (64-bit x86) / ami-07a3e3eda401f8caa (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

macOS Big Sur 11.3.1 - ami-0aab315fd64928e58

The macOS Big Sur AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

macOS Catalina 10.15.7 - ami-050efa637ab09b285

The macOS Catalina AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

macOS Mojave 10.14.6 - ami-09cda95c1a82e9f7b

The macOS Mojave AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

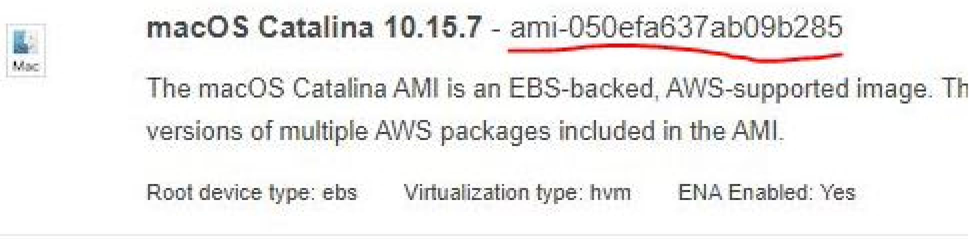
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0ba62214afa52bec7 (64-bit x86) / ami-09f8674883d0ad6b8 (64-bit Arm)

Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Figure 26 : Choosing AMI ID and Properties



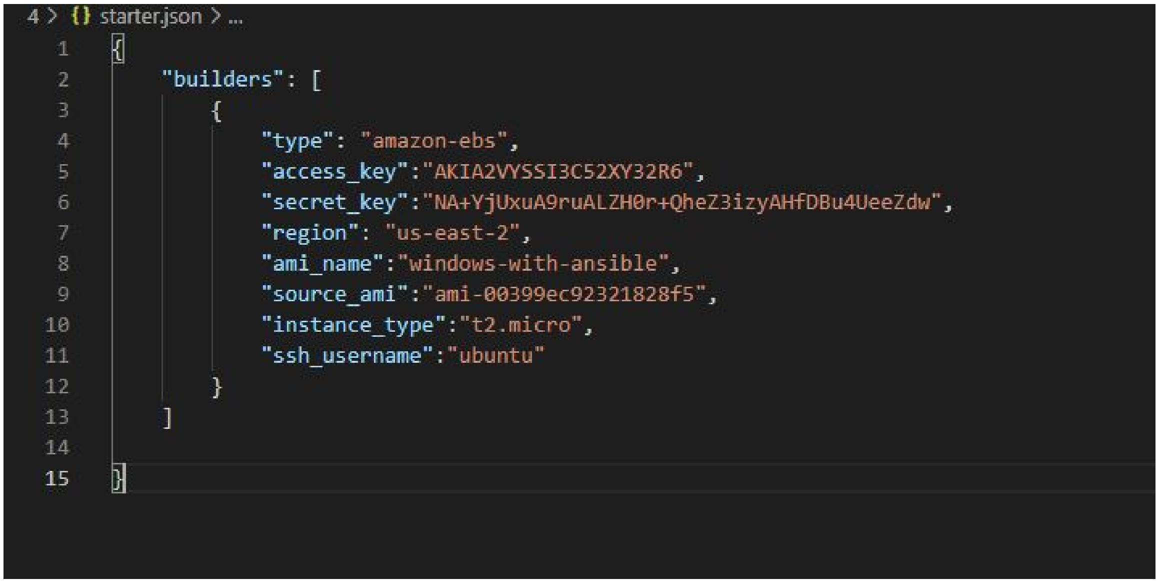
macOS Catalina 10.15.7 - ami-050efa637ab09b285

The macOS Catalina AMI is an EBS-backed, AWS-supported image. Th
versions of multiple AWS packages included in the AMI.

Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

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.

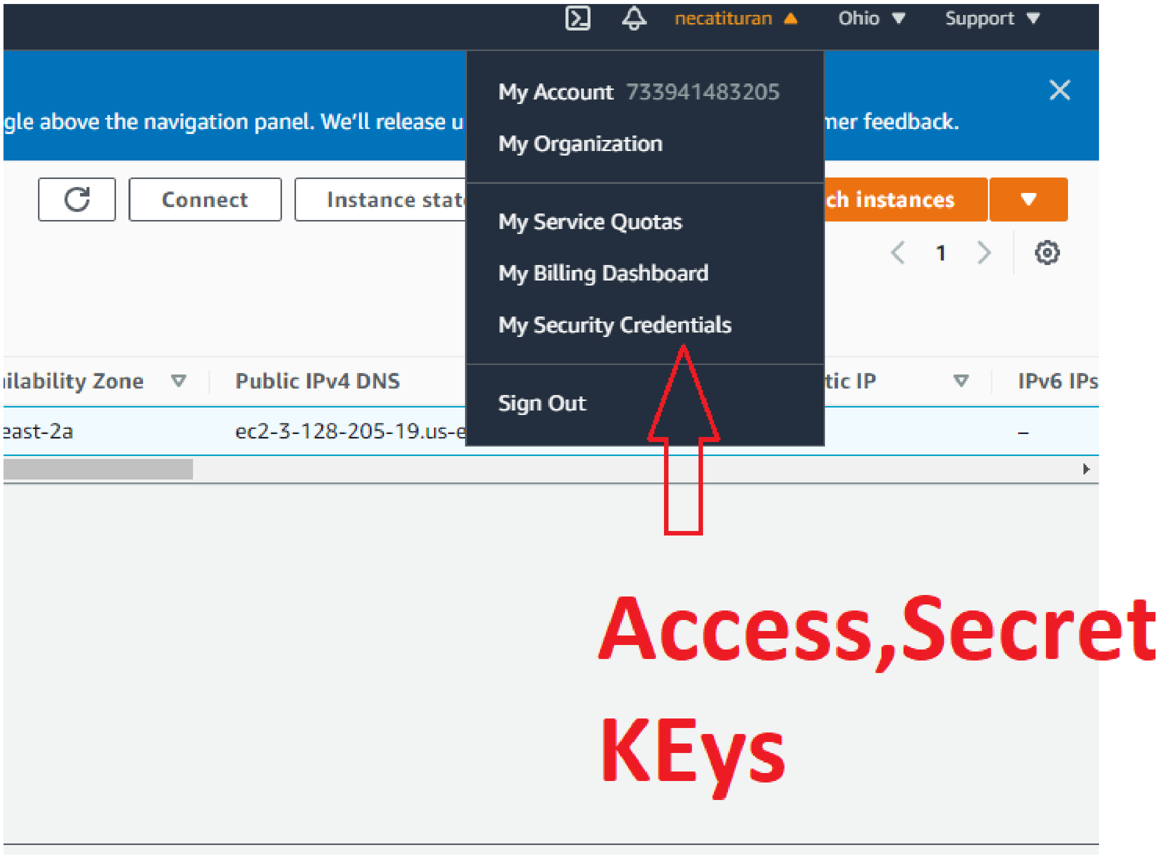
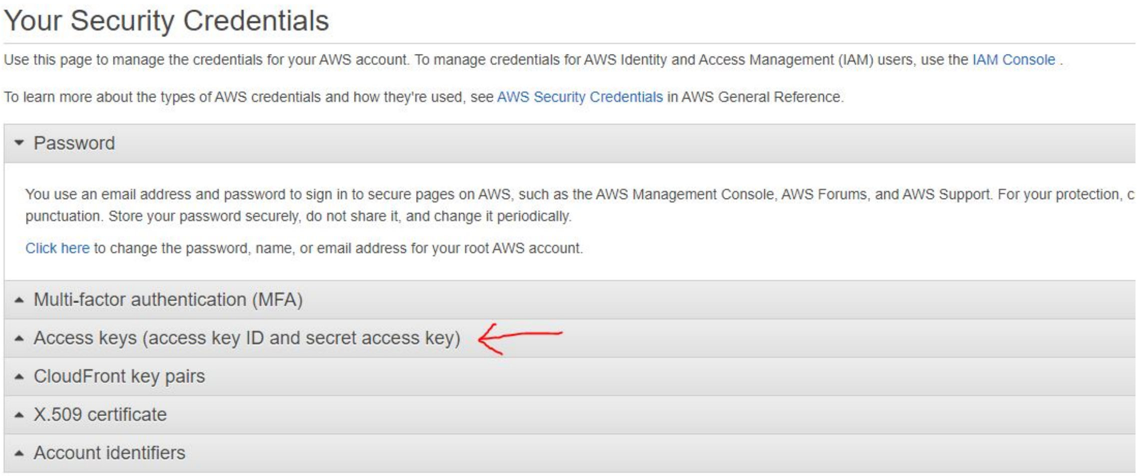


Figure 27: Security Credentials



Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the IAM Console .

To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference.

▾ Password

You use an email address and password to sign in to secure pages on AWS, such as the AWS Management Console, AWS Forums, and AWS Support. For your protection, c punctuation. Store your password securely, do not share it, and change it periodically.

Click here to change the password, name, or email address for your root AWS account.

▴ Multi-factor authentication (MFA)

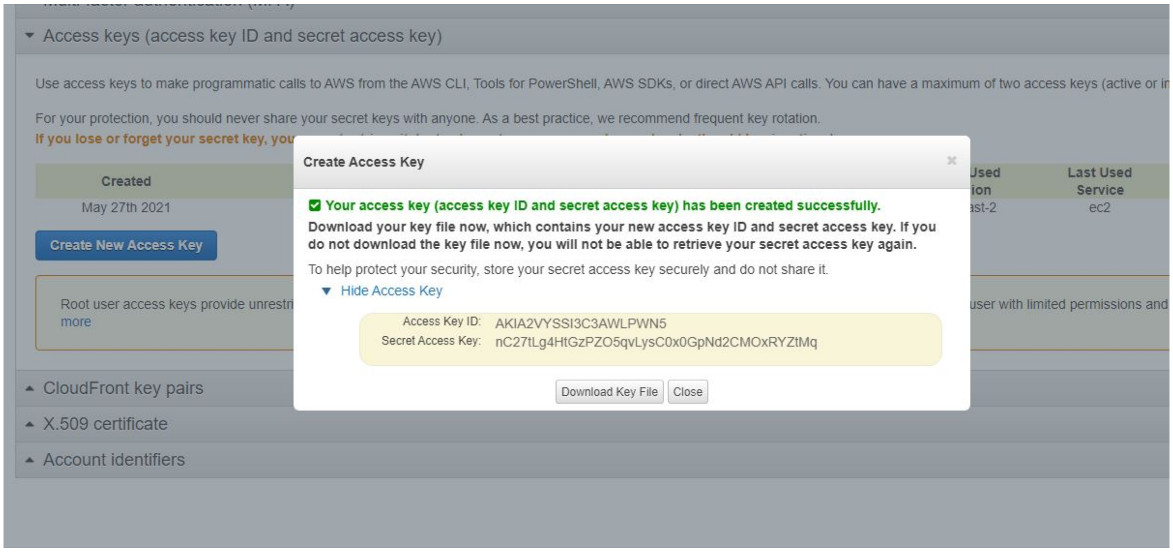
▴ Access keys (access key ID and secret access key)

▴ CloudFront key pairs

▴ X.509 certificate

▴ Account identifiers

Figure 28 : Access Keys



Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or in

For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation.

If you lose or forget your secret key, you

Created	Used ion	Last Used Service
May 27th 2021	ast-2	ec2

Root user access keys provide unrestri user with limited permissions and more

Create Access Key

☑ Your access key (access key ID and secret access key) has been created successfully.

Download your key file now, which contains your new access key ID and secret access key. If you do not download the key file now, you will not be able to retrieve your secret access key again.

To help protect your security, store your secret access key securely and do not share it.

▼ Hide Access Key

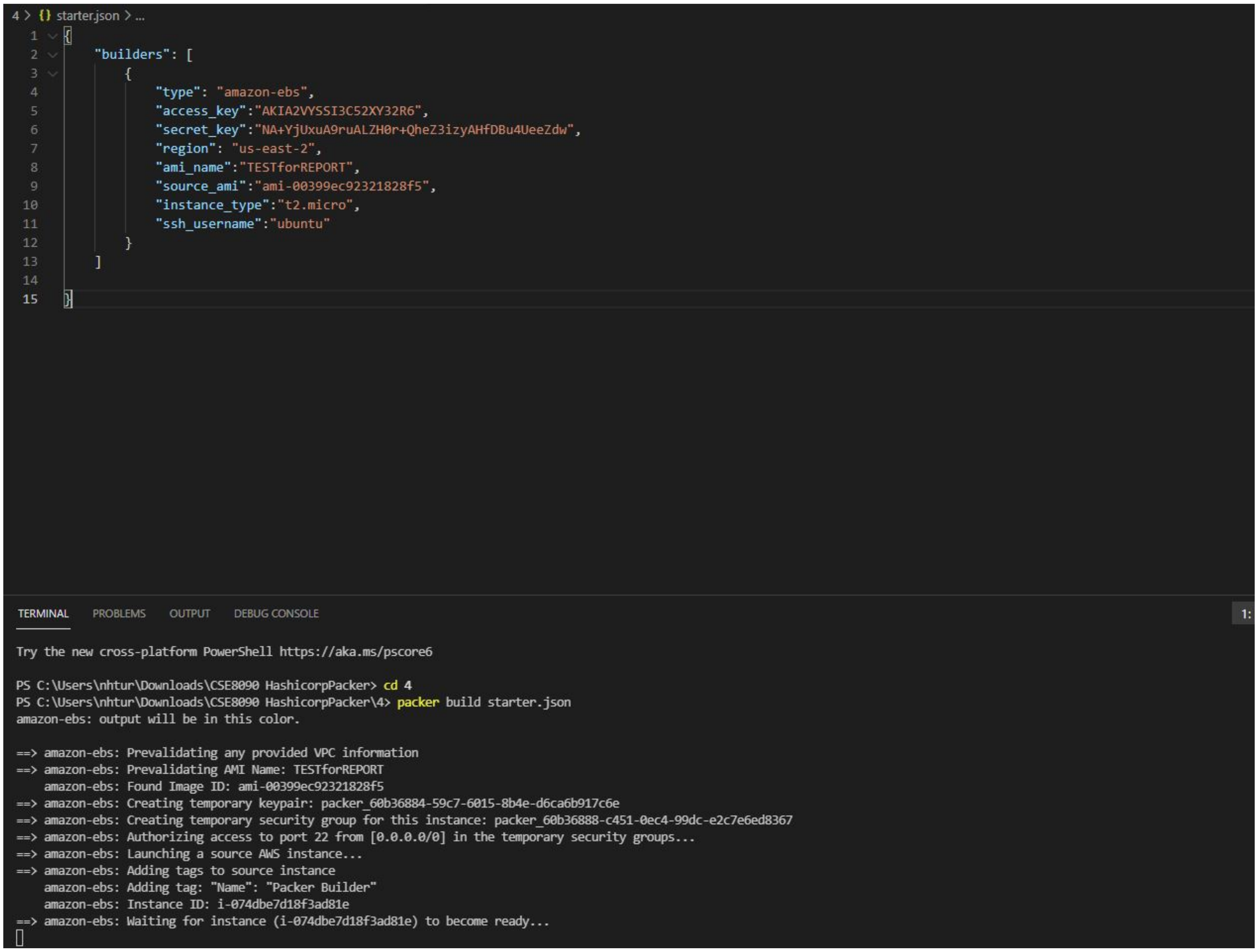
Access Key ID: AKIA2VYSSI3C3AWLPWN5
Secret Access Key: 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

Figure 29 : Downloading Key

Now we can test our first AMI in the VS Code Terminal.

To build our AMI on the terminal, we used this basic command.

```
packer build test.json
```

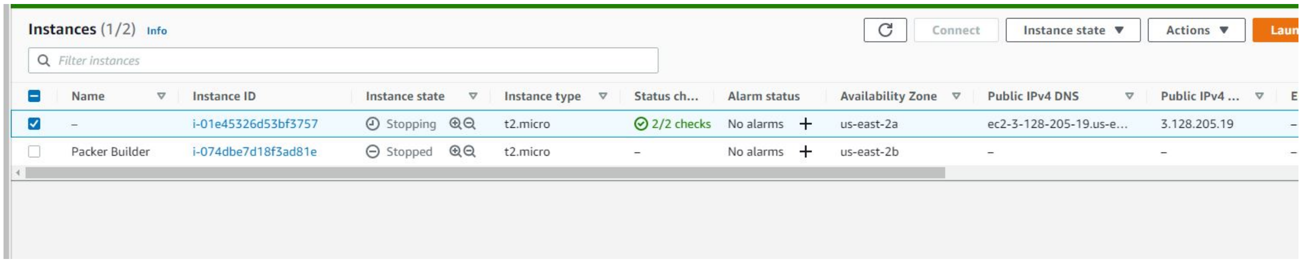


```json
{
    "builders": [
        {
            "type": "amazon-ebs",
            "access_key":"AKIA2VYSSI3C52XY32R6",
            "secret_key":"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",
            "region": "us-east-2",
            "ami_name":"TESTforREPORT",
            "source_ami":"ami-00399ec92321828f5",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

}
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\nhtur\Downloads\CSE8090 HashicorpPacker> cd 4
PS C:\Users\nhtur\Downloads\CSE8090 HashicorpPacker\4> packer build starter.json
amazon-ebs: output will be in this color.

==> amazon-ebs: Prevalidating any provided VPC information
==> amazon-ebs: Prevalidating AMI Name: TESTforREPORT
    amazon-ebs: Found Image ID: ami-00399ec92321828f5
==> amazon-ebs: Creating temporary keypair: packer_60b36884-59c7-6015-8b4e-d6ca6b917c6e
==> amazon-ebs: Creating temporary security group for this instance: packer_60b36888-c451-0ec4-99dc-e2c7e6ed8367
==> amazon-ebs: Authorizing access to port 22 from [0.0.0.0/0] in the temporary security groups...
==> amazon-ebs: Launching a source AWS instance...
==> amazon-ebs: Adding tags to source instance
    amazon-ebs: Adding tag: "Name": "Packer Builder"
    amazon-ebs: Instance ID: i-074dbe7d18f3ad81e
==> amazon-ebs: Waiting for instance (i-074dbe7d18f3ad81e) to become ready...
```

Figure 30 : Packer Basics : Build

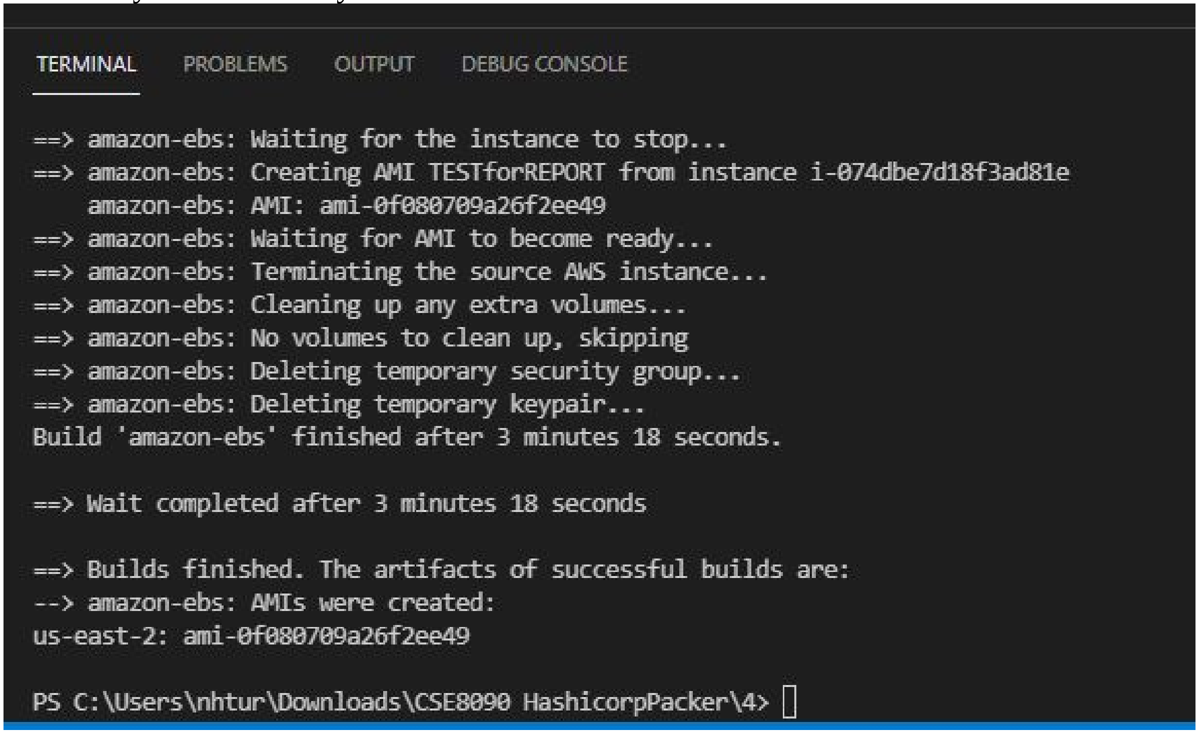
Packer will create and AMI and for that purpose it will create an instance and then it will stopped that immediately after creating.



☐	Name	Instance ID	Instance state	Instance type	Status ch...	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	E
☑	–	i-01e45326d53bf3757	Stopping	t2.micro	2/2 checks	No alarms +	us-east-2a	ec2-3-128-205-19.us-e...	3.128.205.19	–
☐	Packer Builder	i-074dbe7d18f3ad81e	Stopped	t2.micro	–	No alarms +	us-east-2b	–	–	–

Figure 31 : Packer Builder Seen on AWS

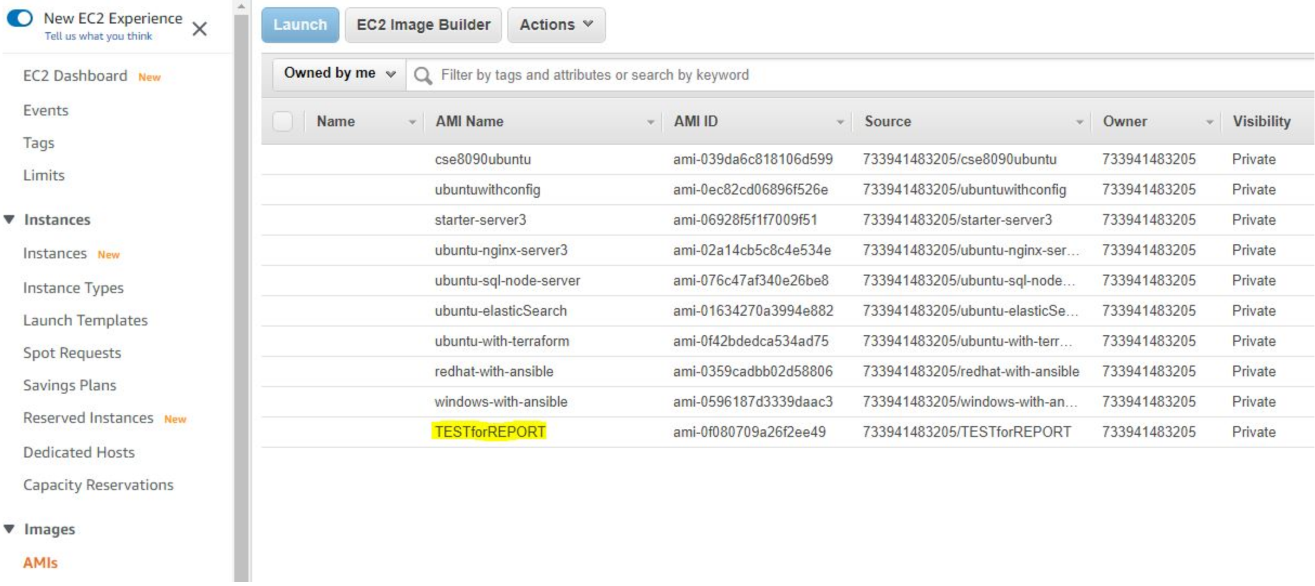
AMI successfully created in 25 seconds from terminal by remote access using only Access keys and secret keys.



```
==> amazon-ebs: Waiting for the instance to stop...
==> amazon-ebs: Creating AMI TESTforREPORT from instance i-074dbe7d18f3ad81e
    amazon-ebs: AMI: ami-0f080709a26f2ee49
==> amazon-ebs: Waiting for AMI to become ready...
==> amazon-ebs: Terminating the source AWS instance...
==> amazon-ebs: Cleaning up any extra volumes...
==> amazon-ebs: No volumes to clean up, skipping
==> amazon-ebs: Deleting temporary security group...
==> amazon-ebs: Deleting temporary keypair...
Build 'amazon-ebs' finished after 3 minutes 18 seconds.

==> Wait completed after 3 minutes 18 seconds

==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs: AMIs were created:
us-east-2: ami-0f080709a26f2ee49

PS C:\Users\nhtur\Downloads\CSE8090 HashicorpPacker\4> 
```

Figure 32 : AMI Created

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



Name	AMI Name	AMI ID	Source	Owner	Visibility
	cse8090ubuntu	ami-039da6c818106d599	733941483205/cse8090ubuntu	733941483205	Private
	ubuntuwithconfig	ami-0ec82cd06896f526e	733941483205/ubuntuwithconfig	733941483205	Private
	starter-server3	ami-06928f5f1f7009f51	733941483205/starter-server3	733941483205	Private
	ubuntu-nginx-server3	ami-02a14cb5c8c4e534e	733941483205/ubuntu-nginx-ser…	733941483205	Private
	ubuntu-sql-node-server	ami-076c47af340e26be8	733941483205/ubuntu-sql-node…	733941483205	Private
	ubuntu-elasticSearch	ami-01634270a3994e882	733941483205/ubuntu-elasticSe…	733941483205	Private
	ubuntu-with-terraform	ami-0f42bdedca534ad75	733941483205/ubuntu-with-terr…	733941483205	Private
	redhat-with-ansible	ami-0359cadbb02d58806	733941483205/redhat-with-ansible	733941483205	Private
	windows-with-ansible	ami-0596187d3339daac3	733941483205/windows-with-an…	733941483205	Private
	TESTforREPORT	ami-0f080709a26f2ee49	733941483205/TESTforREPORT	733941483205	Private

Figure 33 : AMI Seen on AWS

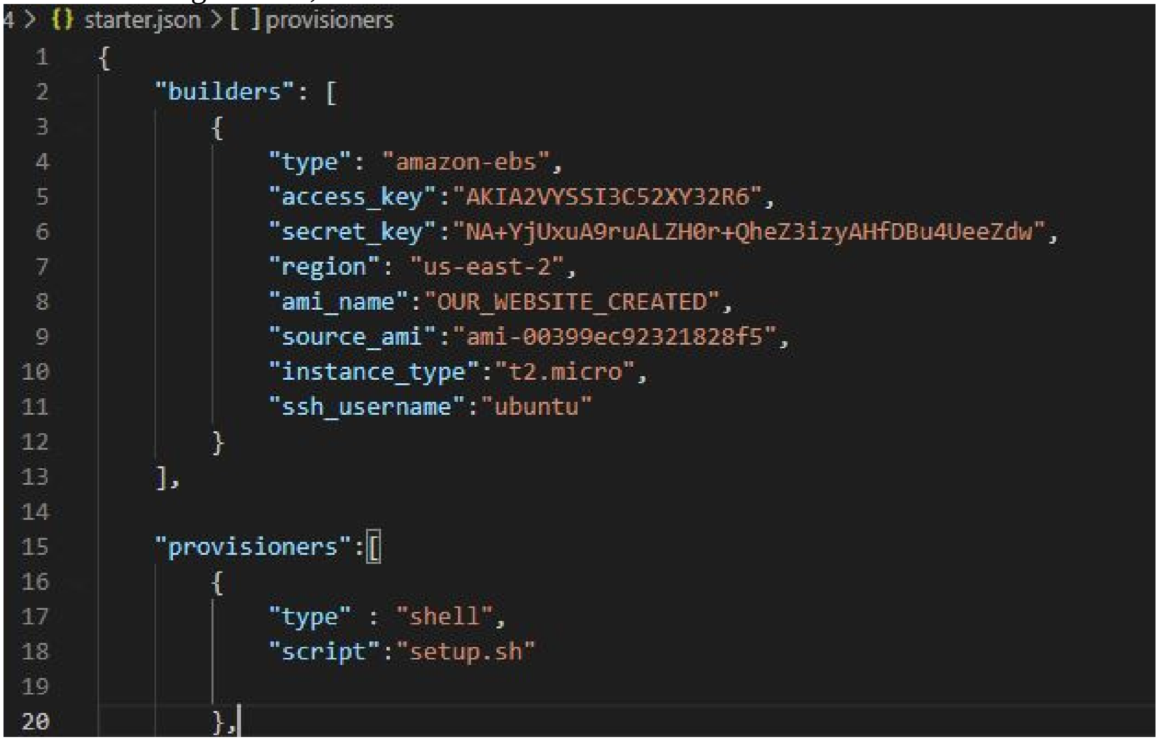
Now we can go further and by only using Shell commands, we can create a directory for our WebSite in the AWS cloud.

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.

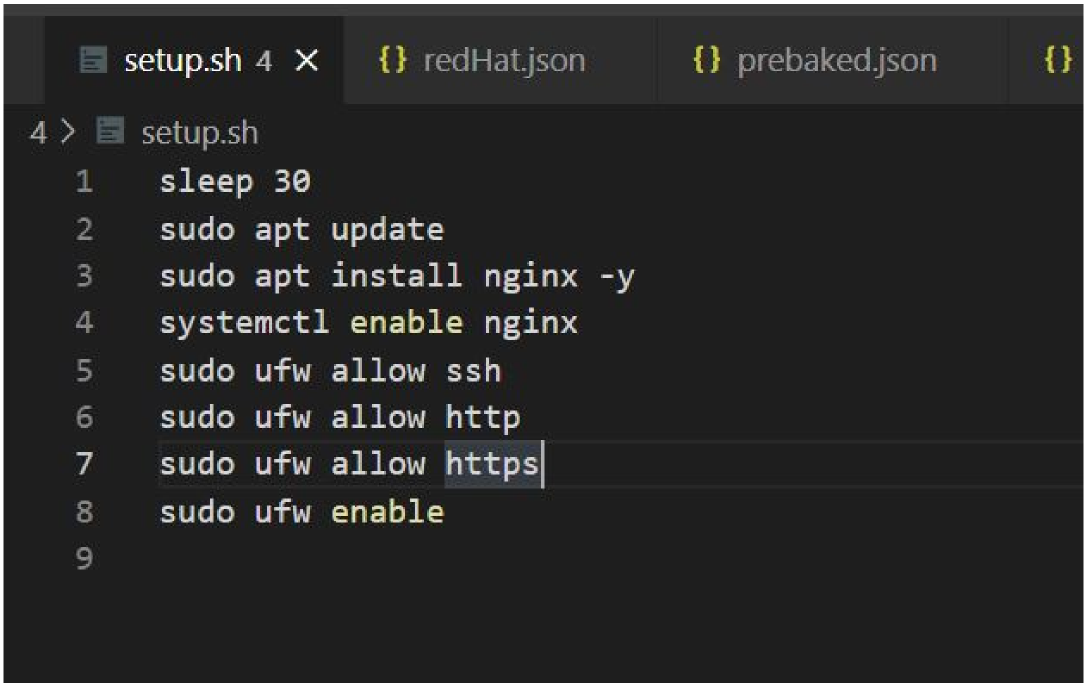


```
{
    "builders": [
        {
            "type": "amazon-ebs",
            "access_key":"AKIA2VYSSI3C52XY32R6",
            "secret_key":"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",
            "region": "us-east-2",
            "ami_name":"OUR_WEBSITE_CREATED",
            "source_ami":"ami-00399ec92321828f5",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

    "provisioners":[
        {
            "type" : "shell",
            "script":"setup.sh"

        },
```

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.

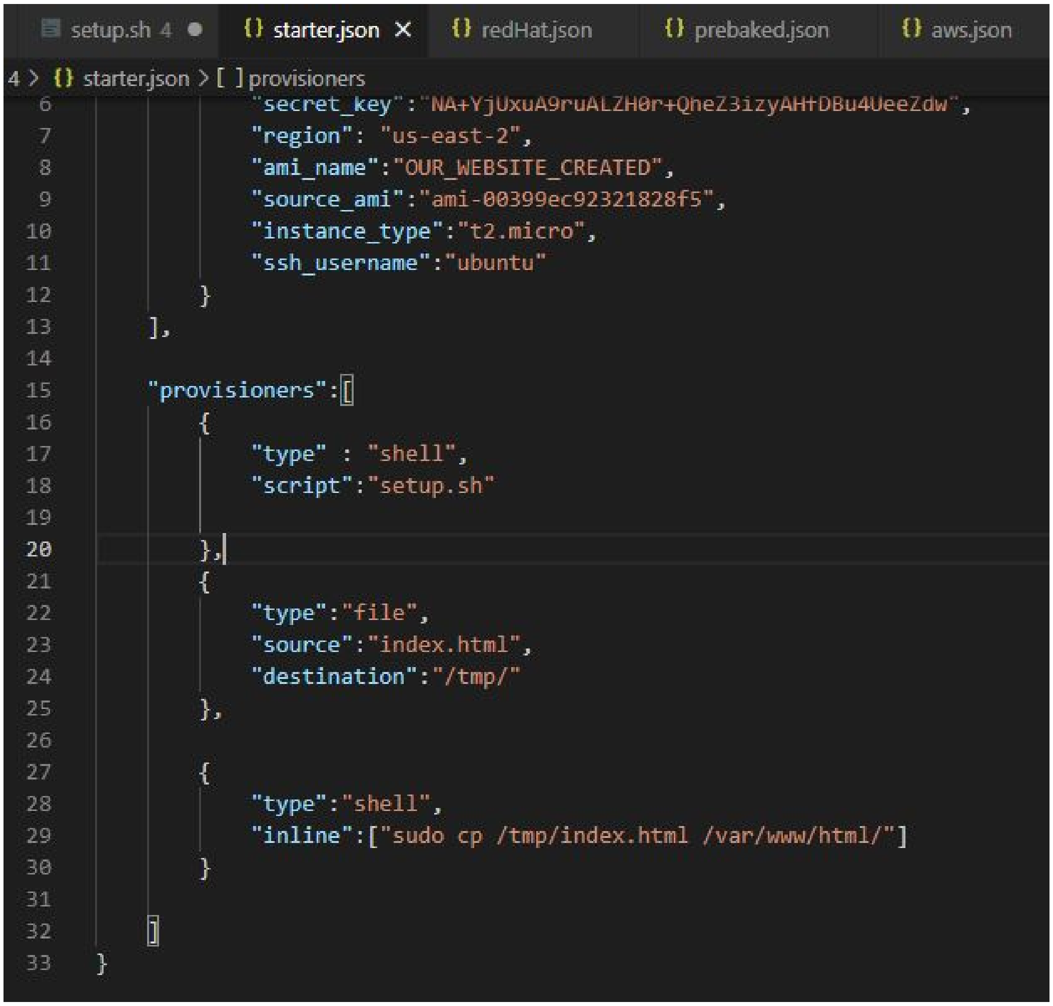
We need NGINX server to be configured first to create a website on AWS. We also need to update the system for this purpose and to make it publicly reachable we need HTTP and HTTPS enabling.



```
sleep 30
sudo apt update
sudo apt install nginx -y
systemctl enable nginx
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
sudo ufw enable
```

Figure 34 : Shell Commands for NGINX Server

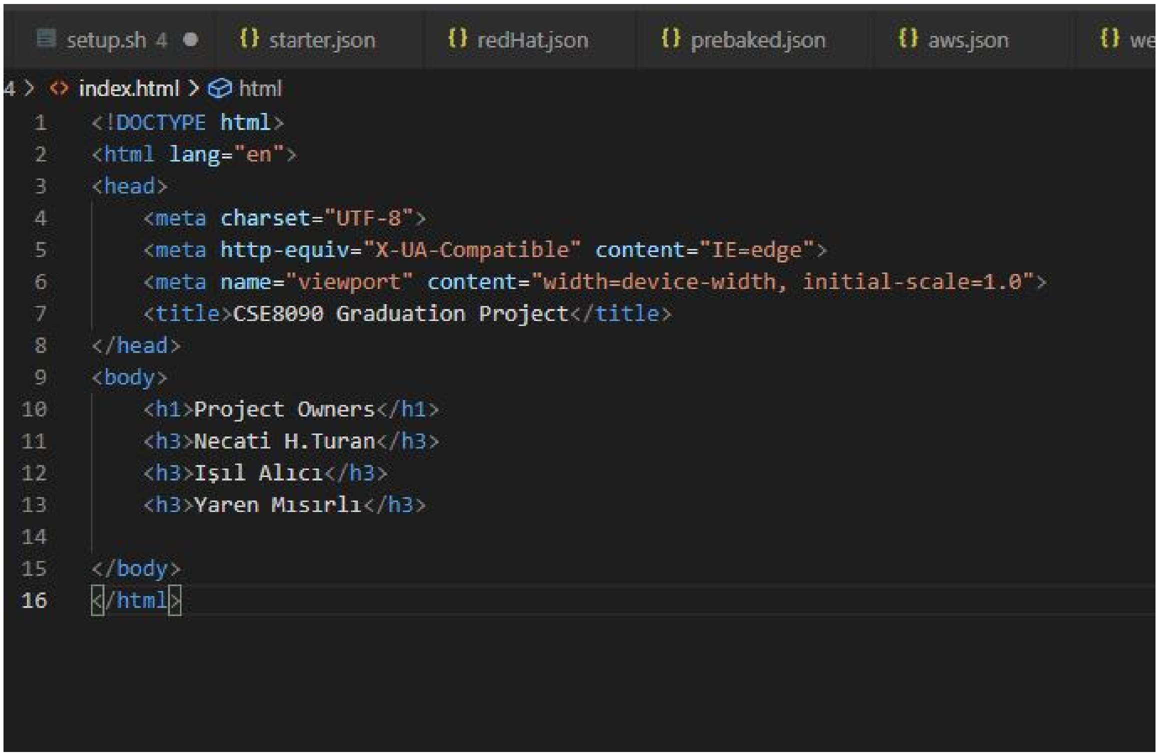
After creating the setup.SH file, we need to attach and make it run in AWS by using packer’s provisioners.



```
                "secret_key":"NA+YjUxuA9ruALZH0r+QheZ3izyAH+DBu4UeeZdw",
                "region": "us-east-2",
                "ami_name":"OUR_WEBSITE_CREATED",
                "source_ami":"ami-00399ec92321828f5",
                "instance_type":"t2.micro",
                "ssh_username":"ubuntu"
            }
        ],

        "provisioners":[
            {
                "type" : "shell",
                "script":"setup.sh"

            },
            {
                "type":"file",
                "source":"index.html",
                "destination":"/tmp/"
            },

            {
                "type":"shell",
                "inline":["sudo cp /tmp/index.html /var/www/html/"]
            }

        ]
    }
```

Figure 35 : Sudo Copy Directory

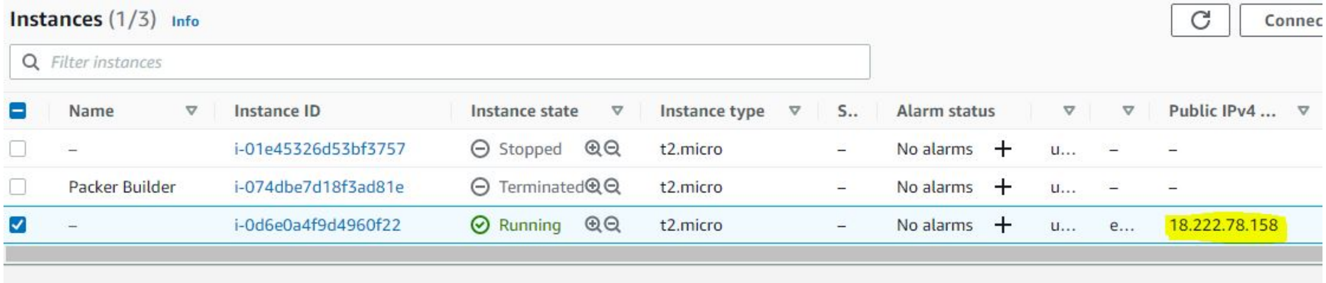
We copy our index.html file to the server by using Sudo privilege from var/www/ to tmp directory.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSE8090 Graduation Project</title>
</head>
<body>
    <h1>Project Owners</h1>
    <h3>Necati H.Turan</h3>
    <h3>Işıl Alıcı</h3>
    <h3>Yaren Mısırlı</h3>

</body>
</html>
```

Figure 36 : Simple HTML File with NGINX

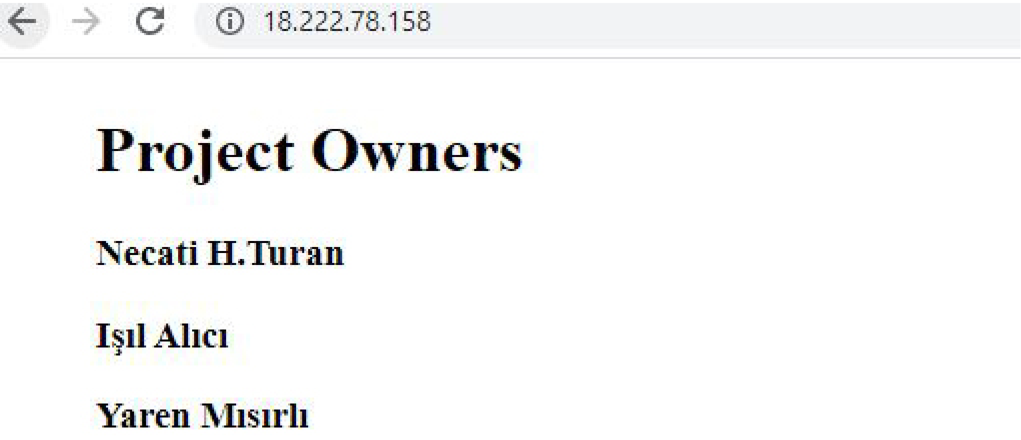
Here is our AMIs public IP address, it’s running on the our AWS account.



	Name	Instance ID	Instance state	Instance type	S..	Alarm status			Public IPv4 ...
☐	–	i-01e45326d53bf3757	Stopped	t2.micro	–	No alarms +	u...	–	–
☐	Packer Builder	i-074dbe7d18f3ad81e	Terminated	t2.micro	–	No alarms +	u...	–	–
☑	–	i-0d6e0a4f9d4960f22	Running	t2.micro	–	No alarms +	u...	e...	18.222.78.158

Figure 37 : Public IP

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



Project Owners

Necati H.Turan

Işıl Alıcı

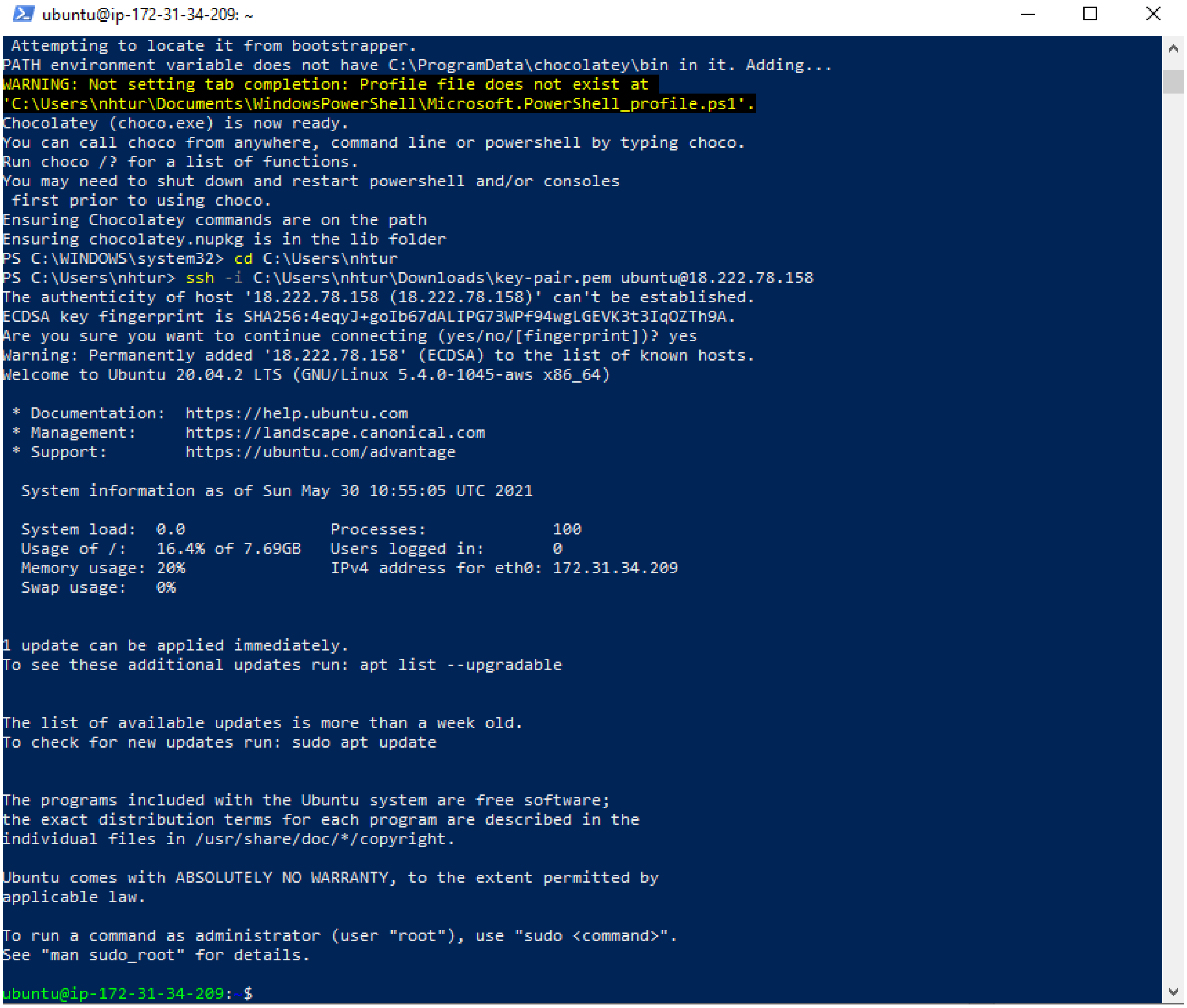
Yaren Mısırlı

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
```

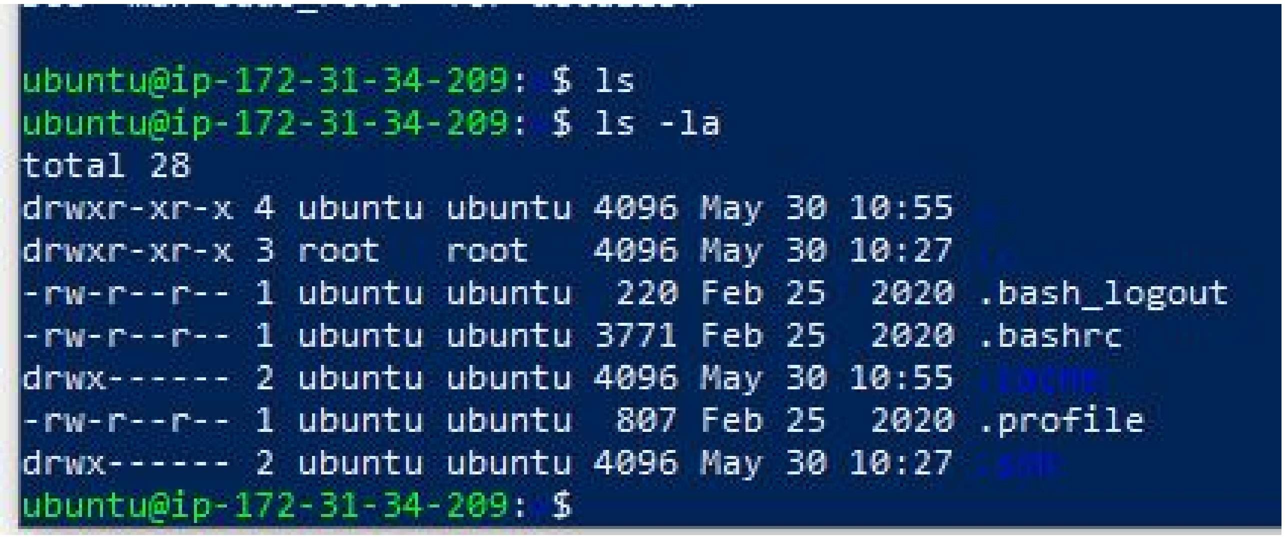


ubuntu@ip-172-31-34-209: ~

```
 Attempting to locate it from bootstrapper.
PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding...
WARNING: Not setting tab completion: Profile file does not exist at
'C:\Users\nhtur\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'.
Chocolatey (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart powershell and/or consoles
 first prior to using choco.
Ensuring Chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
PS C:\WINDOWS\system32> cd C:\Users\nhtur
PS C:\Users\nhtur> ssh -i C:\Users\nhtur\Downloads\key-pair.pem ubuntu@18.222.78.158
The authenticity of host '18.222.78.158 (18.222.78.158)' can't be established.
ECDSA key fingerprint is SHA256:4eqyJ+goIb67dALIPG73WPf94wgLGEVK3t3IqOZTh9A.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '18.222.78.158' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1045-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun May 30 10:55:05 UTC 2021

  System load:  0.0               Processes:             100
  Usage of /:   16.4% of 7.69GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for eth0: 172.31.34.209
  Swap usage:   0%


1 update can be applied immediately.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-34-209:~$
```

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.



```
ubuntu@ip-172-31-34-209:~$ ls
ubuntu@ip-172-31-34-209:~$ ls -la
total 28
drwxr-xr-x 4 ubuntu ubuntu 4096 May 30 10:55 .
drwxr-xr-x 3 root   root   4096 May 30 10:27 ..
-rw-r--r-- 1 ubuntu ubuntu  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Feb 25  2020 .bashrc
drwx------ 2 ubuntu ubuntu 4096 May 30 10:55 .cache
-rw-r--r-- 1 ubuntu ubuntu  807 Feb 25  2020 .profile
drwx------ 2 ubuntu ubuntu 4096 May 30 10:27 .ssh
ubuntu@ip-172-31-34-209:~$
```

Figure 40 : DRWXR

4. 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.

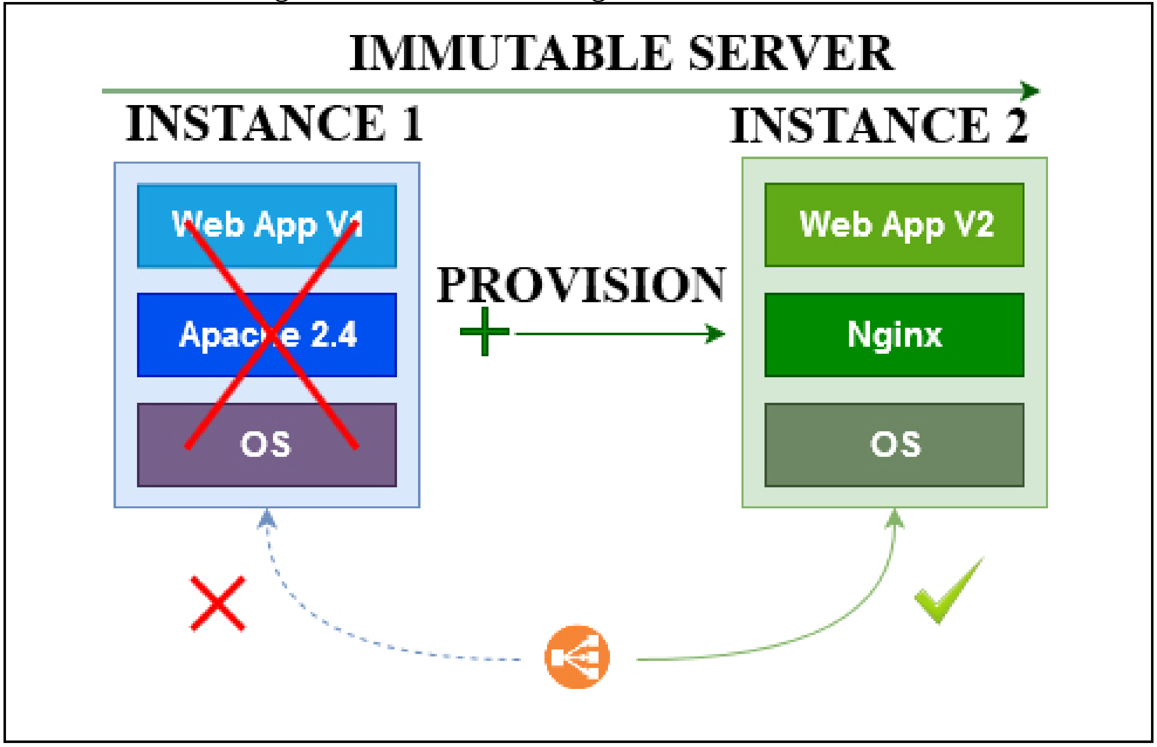


Figure 41:Diagram of Immutable Achieved On This Paper

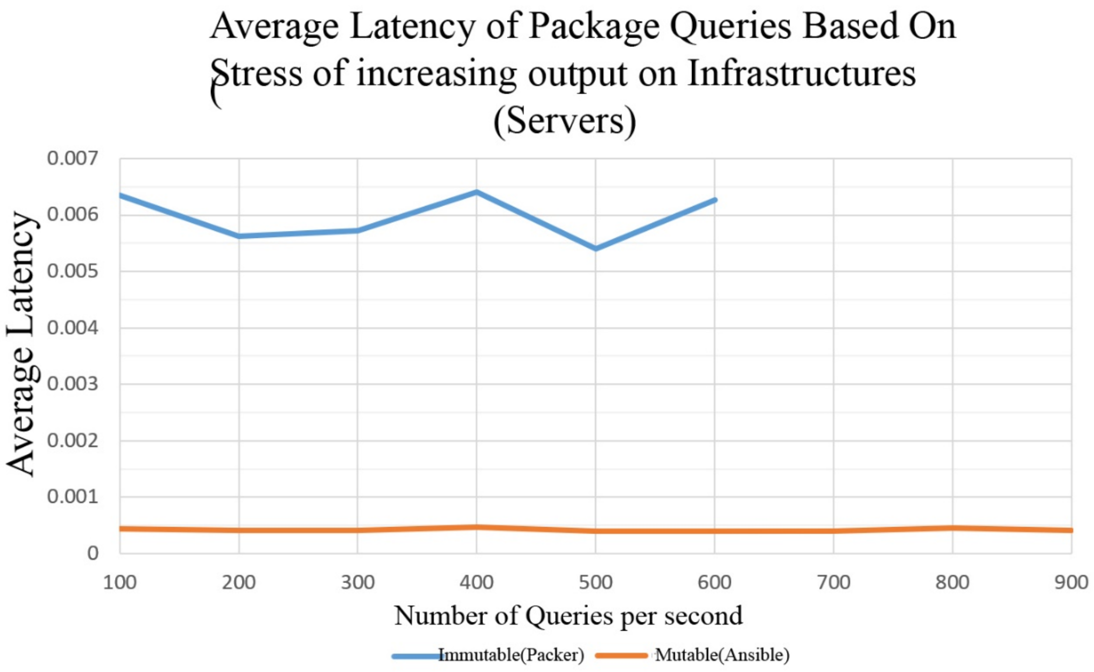


Figure 42 : Benchmark of Packer vs. Opponent

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

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

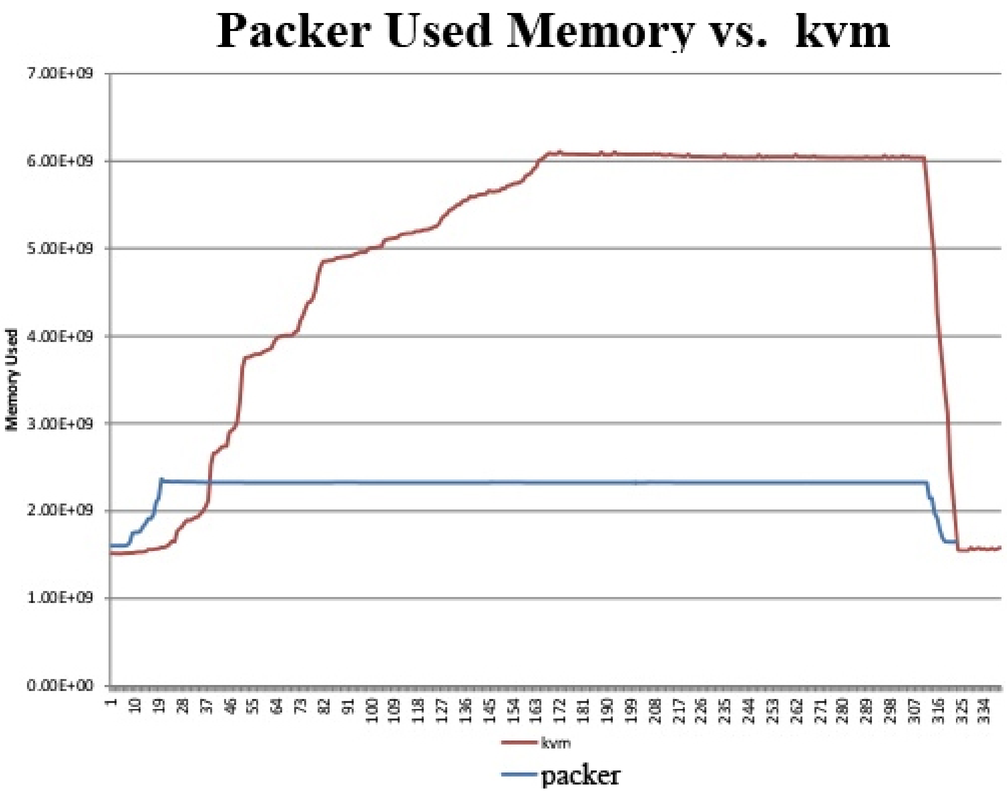


Figure 43:Packer vs. Opponent on Memory Usage

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

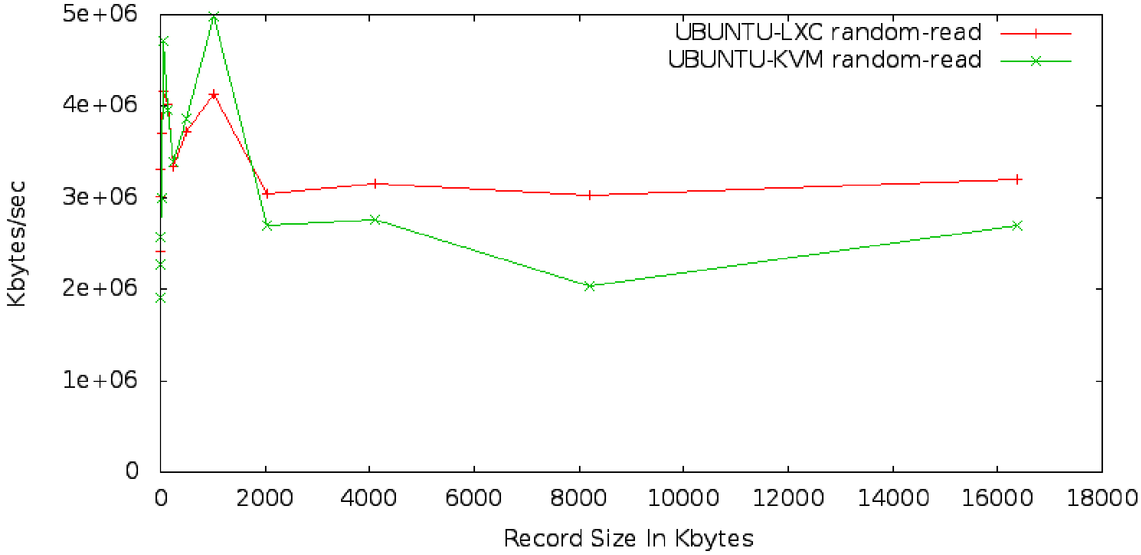


Figure 44: Ubuntu on Cloud Record Sizes

5. 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.

6. 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.

State a brief summary of your interpretations and conclusions regarding the project’s topic. Recommended structure moves from Specific to General;

- Begins with a reiteration of the project topic (tone is emphatic),
- May summarize main points and findings,
- Brings the topic back to some general significance or relevance,
- Finally, provides future directions to this study.

REFERENCES

Every citation made in the body of the project report must appear in the References. Similarly, every item listed in the References must be cited in the body of the report. Follow the APA standard method for citing and listing both the print references and online references. Examples;

[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. <i>J. Comput. Sci. Coll.</i> 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 <i>Proceedings of the 14th Brazilian Symposium on Software Components, Architectures, and Reuse</i> (<i>SBCARS '20</i>). 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] Immutable Infrastructure Calls for Immutable Architecture,Conference Paper 2019, DOI: 10.24251/HICSS.2019.846