Volume: 11 Issue: III Month of publication: March 2023
DOI: https://doi.org/10.22214/ijraset.2023.49333

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue III Mar 2023- Available at www.ijraset.com

Cloud Formation (IaC): Deploying a Containerized Application on Cloud

Shubham More[1], Uddesh Piprewar[2], Vishal Lamsoge[3], Balwesh Puramkar[4], Gayatri Dandhare[5]
[1, 2, 3, 4, 5]B.E. Graduate (iv year), Department of Computer Science and Engineering, NIT, Nagpur

Abstract: This study is a literature review on cloud computing trends as one of the Fastest growing technologies in the computer industry and their benefits and opportunities for all types of organizations. In addition, it addresses the challenges and problems that contribute to increasing the number of customers willing to adopt and use the technology. A mixed research study approach was adopted for the study, that is by collecting and analysing both quantitative and qualitative information within the sane literature review and summarizing the findings of previous (related) studies. Results highlights the current and future trends of cloud computing and exposes readers to the challenges and problems associated with cloud computing. The reviewed literature showed literature showed that the technology is promising and is expected to grow in the future. Researchers have proposed many techniques to address the problems and challenges of cloud computing, such as security and privacy risks, through mobile cloud computing and cloud-computing governance.

I. INTRODUCTION

Like on-premises infrastructures, modern Cloud infrastructures are a tangle of diverse, interdependent components: to work in harmony, instances, storage, load balancers, firewalls, databases, and content delivery networks must be correctly provisioned and configured a historically manual process that’s complex, time-consuming, and error- prone.
Managing your infrastructure with many services can be hard. Creating and managing multiple AWS resources can be challenging and time-consuming. In fact, doing those things could result in spending a whole lot of time managing your AWS resources instead of developing your applications. How can we solve this problem?
AWS Cloud Formation can help. As mentioned, it provides you with a simple way to create and manage a collection of AWS resources by provisioning and updating them in an orderly and predictable way. In simple terms, it allows you to create and model your infrastructure and applications without having to perform actions manually. AWS CloudFormation enables you to manage your complete infrastructure or AWS resources in a text file, or template. A collection of AWS resources is called a stack. AWS resources can be created or updated by using a stack. All the resources you require in an application can be deployed easily using templates. Also, you can reuse your templates to replicate your infrastructure in multiple environments. To make templates reusable, use the parameters, mappings and conditions sections in the template so that you can customize your stacks when you create them.

1) Create a new template or use an existing Cloud Formation template using the JSON or YAML format.
2) Save your code template locally or in an S3 bucket.
3) Use AWS Cloud Formation to build a stack on your template.
4) AWS Cloud Formation constructs and configures the stack resources that you have specified in your template.

II. PROBLEM STATEMENT

AWS Cloud Formation is a service that gives developers and businesses an easy way to create a collection of related AWS and third- party resources, and provision and manage them in an orderly and predictable fashion.
Developers can deploy and update compute, database, and many otherresources in a simple, declarative style that abstracts away the complexity of specific resource APIs. AWS Cloud Formation is designed to allow resource lifecycles to be managed repeatably, predictable, and safely, while allowing for automatic rollbacks, automated state management, and management of resources across accounts and regions.
Recent enhancements and options allow for multiple ways to create resources, including using AWS CDK for coding in higher-level languages, importing existing resources, detecting configuration drift, and a new Registry that makes it easier to createcustom types that inherit many core CloudFormation benefits.
Best practices are recommendations that can help you use AWS CloudFormation more effectively and securely throughout its entire workflow.

Learn how to plan and organize your stacks, create templates that describe your resources and the software applications that run on them, and manage your stacks and their resources. The following best practices are based on real-world experience from current Cloud Formation customers.
Even if you do not initially expect to deploy multiple instances of the same AWS resources, Cloud Formation templates are useful because they ensure that you can scale your environment up quickly when the time comes.
By keeping Cloud Formation templates on hand, you will know that you can add more virtual machine instances or storage space, for example, at a moment's notice if your applications experience increased traffic and you need to scale your environment up.
Alternatively, when demand decreases and you want to scale down to save money, you can take some of your deployments offline while still retaining the ability to redeploy them quickly using Cloud Formation when demand increases.

III. LITERATURE SURVEY

A. Secure Outsourcing Of Scientific Computations

AUTHORS :- M. J. Atallah, K. N.
Pantazopoulos, J. R. Rice, and E. E. Spafford
Y. Rajesh Babu
Assistant professor, Department of CSE Priyadarshini Institute of Technology and Science for women, Chintalapudi, India
We investigate the outsourcing of numerical and scientific computations using the following framework : A customer who needs computations done but lacks the computational resources (computing power, appropriate software, or programming expertise) to do these locally, would like to use an external agent to perform these computations. This currently arises in many practical situations, including the financial services and petroleum services industries.
The outsourcing is secure if it is done without revealing to the external agent either the actual data or the actual answer to the computations.
The general idea is for the customer to do some carefully designed local pre-processing (disguising) of the problem and/or data before sending it to the agent, and also some local postprocessing of the answer returned to extract the true answer. The disguise process should be as lightweight as possible, e.g., take time proportional to the size of the input and answer. The disguise pre-processing that that the customer performs locally to "hide" the real computation can change the numerical properties of the computational performance.
We present a frame work for disguising scientific computations and discuss their costs, numerical properties, and levels of security. These disguise techniques can be embedded in a very high level, easy to-use system (problem solving environment) that hides their complexity.

B. Provable Data Possession at Untrusted Stores

AUTHORS: G. Attendees et al We introduce a model for provable data possession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it.
The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs.
The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. Thus, the PDP model for remote data checking supports large data sets in widely- distributed storage system.

IV. METHODOLOGY

The AWS: Api Gateway: Method resource creates API Gateway methods that define the parameters and body that clients must send in their requests.

Syntax
To declare this entity in your AWS Cloud Formation template, use the following syntax:

JSON

```
{                                                              }

  "Type" :
"AWS::ApiGateway::Method", "Properties" : {

      "ApiKeyRequired" : Boolean,

      "AuthorizationScopes" : [ String, ... ],

      "AuthorizationType" : String, "AuthorizerId" : String, "HttpMethod" :

      String, "Integration" : Integration,

      "MethodResponses" : [
MethodResponse, ... ],

      "OperationName" : String,

      "RequestModels" : {Key : Value, ...},

      "RequestParameters" : {Key : Value, ...},

      "RequestValidatorId" :
String,

      "ResourceId" : String,

      "RestApiId" : String

    }
```

YAML

```
Type: AWS::ApiGateway::Method Properties:

  ApiKeyRequired: Boolean

  AuthorizationScopes:

    - String

  AuthorizationType: String AuthorizerId: String HttpMethod: String

  Integration:

    Integration

  MethodResponses:

    - MethodResponse OperationName: String RequestModels:

    Key : Value

  RequestParameters:
    Key : Value

  RequestValidatorId: String

  ResourceId: String
```

V. FLOWCHART

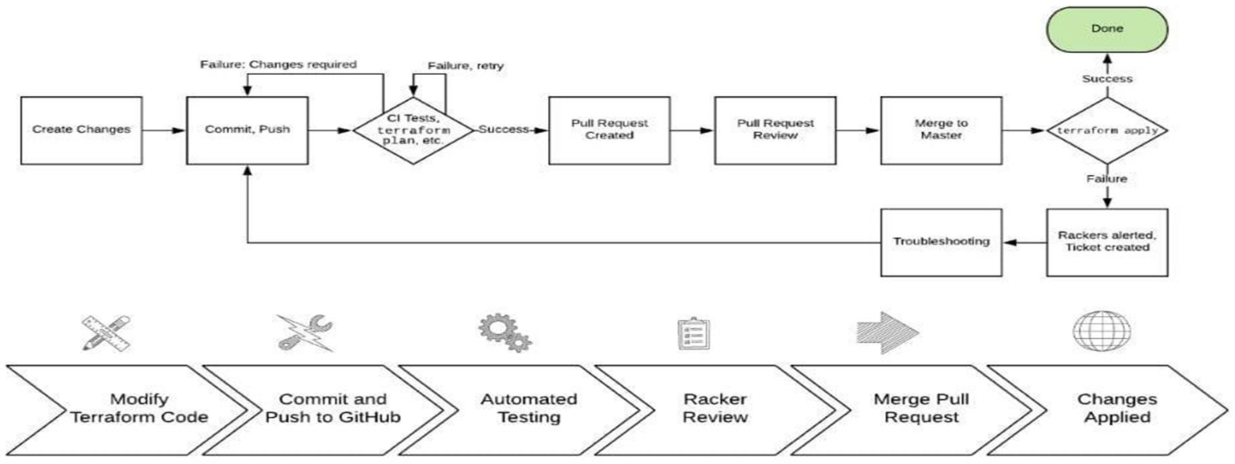


Fig. Cloud Formation Workflow

VI. WHY IS THE PARTICULAR TOPIC CHOSEN

Developers can deploy and update compute, database, and many other resources in a simple, declarative style that abstracts away the complexity of specific resource APIs. AWS CloudFormation is designed to allow resource lifecycles to be managed repeatably, predictable, and safely, while allowing for automatic rollbacks, automated state management, and management of resources across accounts and regions.

Recent enhancements and options allow for multiple ways to create resources, including using AWS CDK for coding in higher-level languages, importing existing resources, detecting configuration drift, and a new Registry that makes it easier to createcustom types that inherit many core Cloud Formation benefits.

AWS Cloud Formation is a servicethat helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and Cloud Formation takes care of provisioning and configuring those resources for you.

You don't need to individually create and configure AWS resources and figure out what's dependent on what; Cloud Formation handles that. The following scenarios demonstrate how Cloud Formation can help.

It is worth noting that Cloud Formation is not the only way to configure and deploy services on AWS. You can handle these processes manually using the AWS command-line interface, API, or Web console.

Manual provisioning is the approach that teams typically take when they are just getting started with AWS and learning how to deploy services. However, as they scale their environments up in size, many teams quickly realize that they need a solution like CloudFormation to make the deployment process faster and more consistent.

VII. HARDWARE AND SOFTWARE USED

1) *Terra form:* Terraform is an open-source, infrastructure as code, software tool created by HashiCorp. Users define and provide data centre infrastructure usinga declarative configuration language known as HashiCorp configuration Language, or optionally JSON.
2) *Docker:* Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. The service has both freeand premium tiers. The software that hosts the containers is called Docker Engine. It was first started in 2013 and is developed by Docker, Inc.
3) *Gitlab/GitHub Actions (CI/CD):* GitLab CI/CD and GitHub Actions both allow you to create workflows that automatically build, test, publish, release, and deploy code. GitLab CI/CD and GitHub Actions share some similarities I workflow configuration: Workflow configuration files arewritten in YAML and are stored in the code’s repository.
4) *Platform:* A platform is a group of technologies that are used as a base upon which other applications, processes or technologies are developed.
5) *Linux:* Just like Windows, iOS and Mac OS, Linux is an operating system, in fact, one of the most popular platform on the planet, Android, is powered by Linux.
6) *Amazon Web Services (AWS):* Amazon web Services, Inc. is a subsidiary of amazon that provides on- demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you- go basis. These cloud computing web services provide distributed computing processing capacity and software tools via AWS server farms.
7) *Docker Hub (Repository):* Docker Hub is a hosted repository service provided by Docker for finding and sharing container images with your team. Key features include: Private Repositories: Push and pull container images. Automated Builds: Automatically build container images from GitHub and Bitbucket and push them to Docker Hub.
8) *YAML(yet another markup language):* YAML is a human-readable data- serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted. YAML targets many of the same communications applications as Extensible Markup Language but has a minimal syntax which intentionally differs from SGML.
9) *HCL (HashiCorp Configuration Language):* HashiCorp Configuration Language (HCL) is a unique configuration language. It was designed to be used with HashiCorp tools, notably Terraform, but HCL has expanded as a more general configuration language. It's visually similar to JSON with additional data structures and capabilities built-in.
10) *Bash (Bourne Again Shell):* Bash is a Unix shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell. First released in 1989, it has been used as the default login shell for most Linux distributions. Bash was one of the first programs Linus Torvalds ported to Linux, alongside GCC.

VIII. WHAT CONTRIBUTION WOULD THE PROJECT

In this Group project we work on cloud platform AWS. here we have create Virtual Private Cloud (VPC) using terraform Tool with the help of AWS provider.

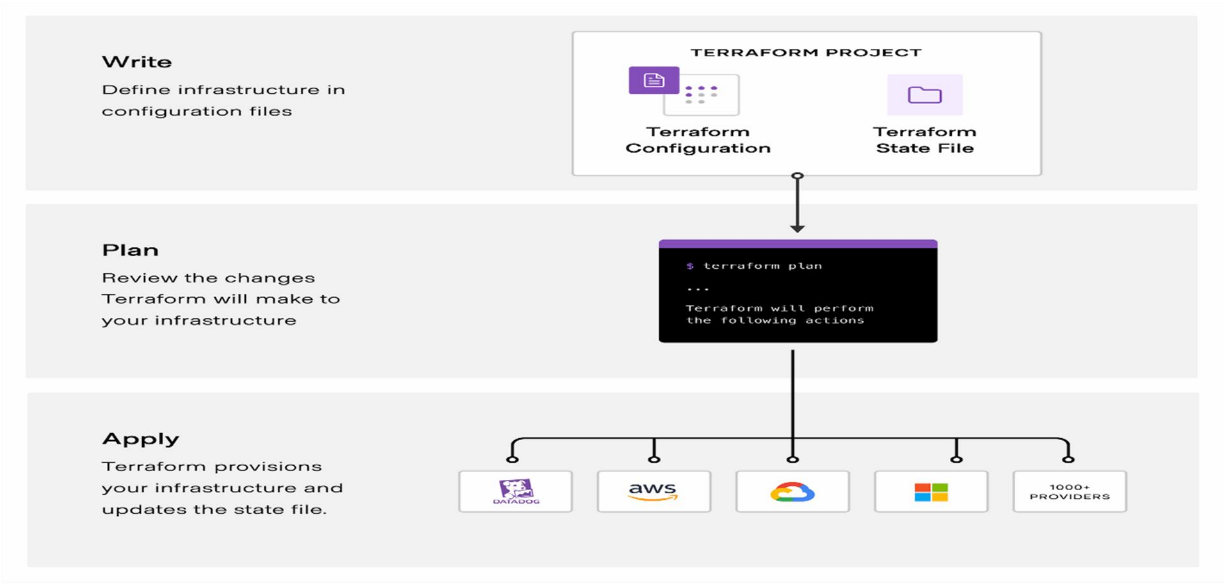
Amazon Virtual Private Cloud (Amazon VPC) provides a logically isolated area of the AWS cloud where you can launch AWS resources in a virtual network that we define.

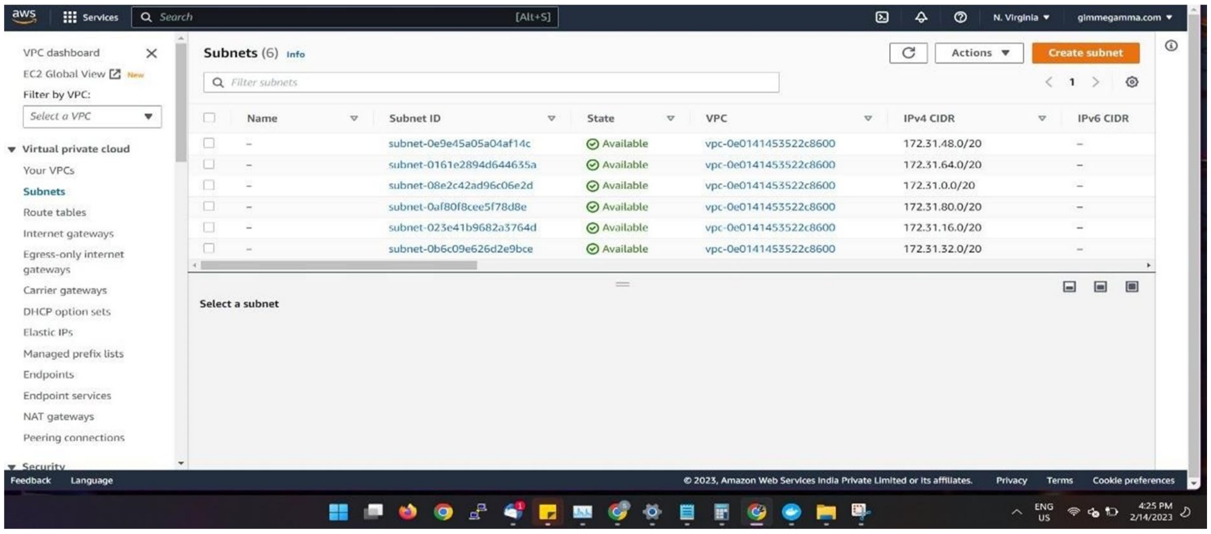
Terraform can provide support with multi- cloud via having a single workflow for every cloud. Various manages of terraform infrastructure could be hosted over Google Cloud Platform, Microsoft Azure, and Amazon Web Services, or on-prem within the private clouds like Cloud Stack, OpenStack, or VMWare vSphere. Terraform considers **IaC** (Infrastructure as Code). So, we need not to be worried about our infrastructure drifting away through the desired configuration.

And deployed existing web application on VPC (Virtual Private Cloud) and we containerized web application using docker platform.

Docker is an open source platform for building, deploying, and managing containerized applications. Learn about containers, how they compare to VMs, and why Docker is so widely adopted and used.

IX. IMAGES


Write
Define infrastructure in configuration files
TERRAFORM PROJECT
Terraform Configuration
Terraform State File
Plan
Review the changes Terraform will make to your infrastructure
$ terraform plan
...
Terraform will perform the following actions
Apply
Terraform provisions your infrastructure and updates the state file.
DATADOG
aws
1000+ PROVIDERS

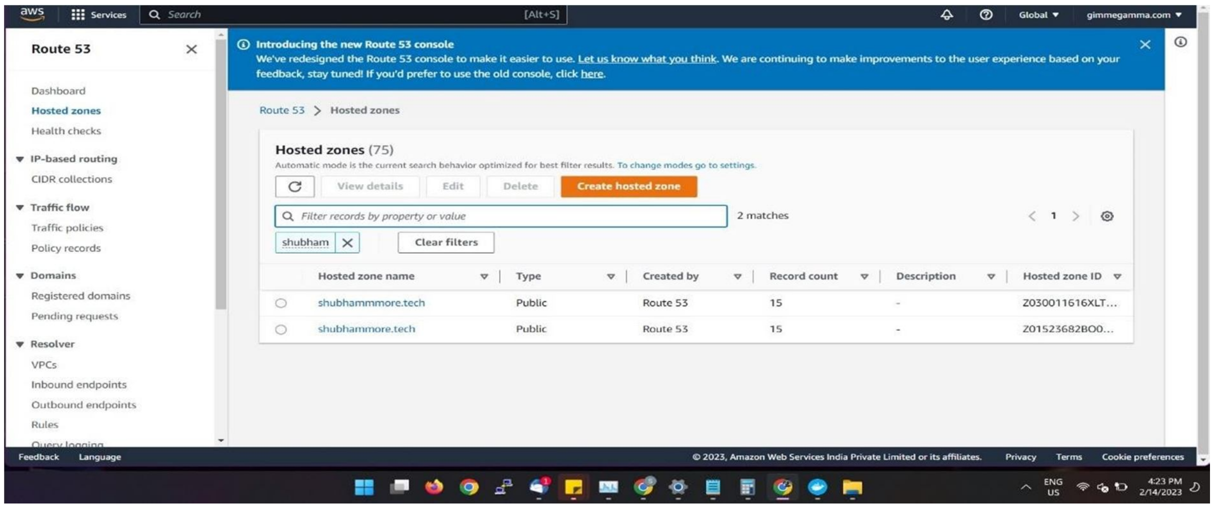


Subnets (6) Info

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR
–	subnet-0e9e45a05a04af14c	Available	vpc-0e0141453522c8600	172.31.48.0/20	–
–	subnet-0161e2894d644635a	Available	vpc-0e0141453522c8600	172.31.64.0/20	–
–	subnet-08e2c42ad96c06e2d	Available	vpc-0e0141453522c8600	172.31.0.0/20	–
–	subnet-0af80f8cee5f78d8e	Available	vpc-0e0141453522c8600	172.31.80.0/20	–
–	subnet-023e41b9682a3764d	Available	vpc-0e0141453522c8600	172.31.16.0/20	–
–	subnet-0b6c09e626d2e9bce	Available	vpc-0e0141453522c8600	172.31.32.0/20	–

Select a subnet

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue III Mar 2023- Available at www.ijraset.com



Introducing the new Route 53 console
We've redesigned the Route 53 console to make it easier to use. Let us know what you think. We are continuing to make improvements to the user experience based on your feedback, stay tuned! If you'd prefer to use the old console, click here.

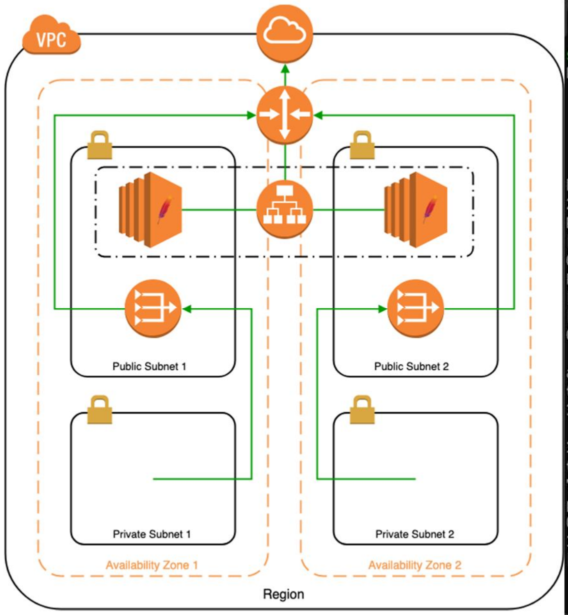
Route 53 > Hosted zones

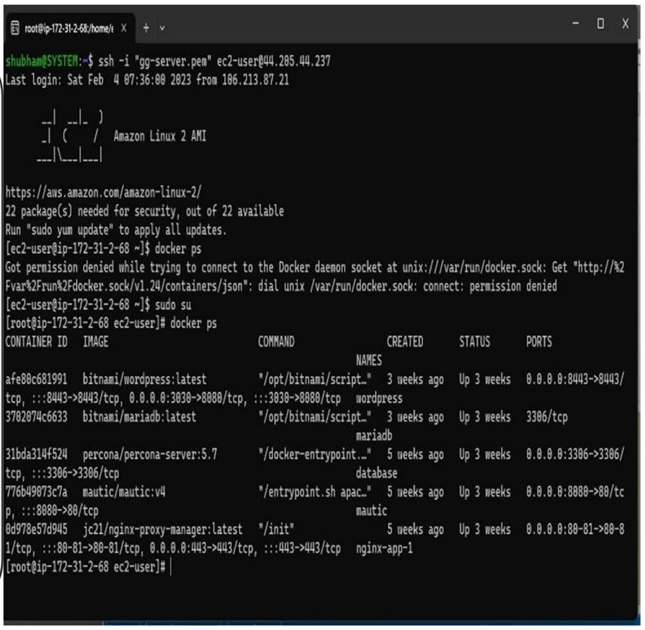
Hosted zones (75)
Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Filter records by property or value — 2 matches — shubham

Hosted zone name	Type	Created by	Record count	Description	Hosted zone ID
shubhammmore.tech	Public	Route 53	15	-	Z030011616XLT...
shubhammore.tech	Public	Route 53	15	-	Z01523682BOO...

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

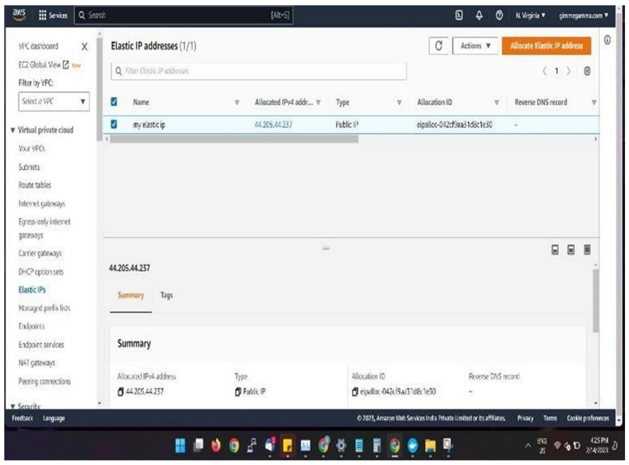




```
shubham@SYSTEM:~$ ssh -i "gg-server.pem" ec2-user@44.205.44.237
Last login: Sat Feb  4 07:36:00 2023 from 106.213.87.21

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
22 package(s) needed for security, out of 22 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-2-68 ~]$ docker ps
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2
Fvar%2Frun%2Fdocker.sock/v1.24/containers/json": dial unix /var/run/docker.sock: connect: permission denied
[ec2-user@ip-172-31-2-68 ~]$ sudo su
[root@ip-172-31-2-68 ec2-user]# docker ps
CONTAINER ID   IMAGE                             COMMAND                  CREATED       STATUS       PORTS
                                                 NAMES
afe80c681991   bitnami/wordpress:latest          "/opt/bitnami/script…"   3 weeks ago   Up 3 weeks   0.0.0.0:8443->8443/
tcp, :::8443->8443/tcp, 0.0.0.0:3030->8080/tcp, :::3030->8080/tcp   wordpress
3702074c6633   bitnami/mariadb:latest            "/opt/bitnami/script…"   3 weeks ago   Up 3 weeks   3306/tcp
                                                 mariadb
31bda314f524   percona/percona-server:5.7        "/docker-entrypoint.…"   5 weeks ago   Up 3 weeks   0.0.0.0:3306->3306/
tcp, :::3306->3306/tcp                           database
776b49073c7a   mautic/mautic:v4                  "/entrypoint.sh apac…"   5 weeks ago   Up 3 weeks   0.0.0.0:8080->80/tc
p, :::8080->80/tcp                               mautic
0d978e57d945   jc21/nginx-proxy-manager:latest   "/init"                  5 weeks ago   Up 3 weeks   0.0.0.0:80-81->80-8
1/tcp, :::80-81->80-81/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp   nginx-app-1
[root@ip-172-31-2-68 ec2-user]# |
```



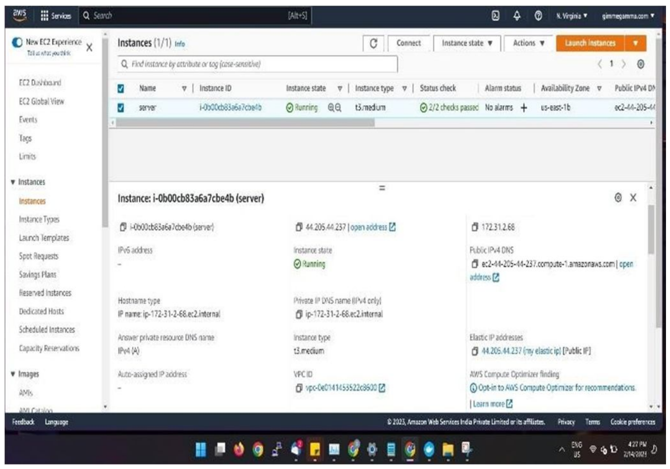
Elastic IP addresses (1/1)

Name	Allocated IPv4 addr...	Type	Allocation ID	Reverse DNS record
my elastic ip	44.205.44.237	Public IP	eipalloc-042cf9aa31d8c1e30	-

44.205.44.237

Summary

Allocated IPv4 address	Type	Allocation ID	Reverse DNS record
44.205.44.237	Public IP	eipalloc-042cf9aa31d8c1e30	-



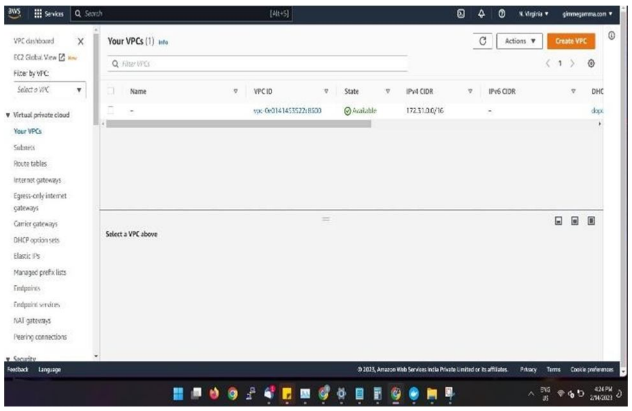
Instances (1/1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
server	i-0b00cb83a6a7cbe4b	Running	t3.medium	2/2 checks passed	No alarms	us-east-1b	ec2-44-205-44

Instance: i-0b00cb83a6a7cbe4b (server)

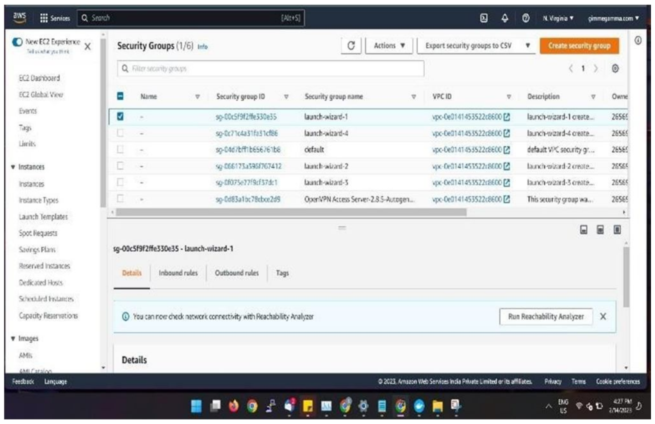
i-0b00cb83a6a7cbe4b (server) — 44.205.44.237 | open address — 172.31.2.68

IPv6 address: –
Instance state: Running
Public IPv4 DNS: ec2-44-205-44-237.compute-1.amazonaws.com | open address
Hostname type: IP name: ip-172-31-2-68.ec2.internal
Private IP DNS name (IPv4 only): ip-172-31-2-68.ec2.internal
Answer private resource DNS name: IPv4 (A)
Instance type: t3.medium
Elastic IP addresses: 44.205.44.237 (my elastic ip) [Public IP]
Auto-assigned IP address: –
VPC ID: vpc-0e0141453522c8600
AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more



Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHC
-	vpc-0e0141453522c8600	Available	172.31.0.0/16	-	dopt

Select a VPC above



Name	Security group ID	Security group name	VPC ID	Description	Owne
-	sg-00c5f9f2ffe330e35	launch-wizard-1	vpc-0e0141453522c8600	launch-wizard-1 create...	2656
-	sg-0c71c4a31fa31cf86	launch-wizard-4	vpc-0e0141453522c8600	launch-wizard-4 create...	2656
-	sg-04d7bff1b656761b8	default	vpc-0e0141453522c8600	default VPC security gr...	2656
-	sg-066173a396f767412	launch-wizard-2	vpc-0e0141453522c8600	launch-wizard-2 create...	2656
-	sg-0f075e77f9cf37dc1	launch-wizard-3	vpc-0e0141453522c8600	launch-wizard-3 create...	2656
-	sg-0d83a1bc78cbce2d9	OpenVPN Access Server-2.8.5-Autogen...	vpc-0e0141453522c8600	This security group wa...	2656

sg-00c5f9f2ffe330e35 - launch-wizard-1

You can now check network connectivity with Reachability Analyzer

Details

REFERENCES

[1] https://docs.aws.amazon.com/AWSE C2/latest/UserGuide/concepts.html
[2] .https://docs.aws.amazon.com/AWSE C2/latest/UserGuide/AMIs.html
[3] https://docs.docker.com/cloud/aci- integration/
[4] https://docs.docker.com/get- started/resources/
[5] https://developer.hashicorp.com/ter raform/cli/init
[6] https://developer.hashicorp.com/ter raform/docs
[7] https://developer.hashicorp.com/ter raform/cli
[8] https://developer.hashicorp.com/ter raform/cli/commands
[9] https://docs.docker.com/desktop/in stall/linux-install/
[10] .https://docs.aws.amazon.com/cli/l atest/reference/ec2/