

4-28-2023

Infrastructure-as-Code: Automating the Deployment on AWS using Terraform

Srikar Pratap

Follow this and additional works at: <https://scholarworks.gvsu.edu/gradprojects>



Part of the [Computer and Systems Architecture Commons](#), and the [Other Computer Engineering Commons](#)

ScholarWorks Citation

Pratap, Srikar, "Infrastructure-as-Code: Automating the Deployment on AWS using Terraform" (2023).
Culminating Experience Projects. 299.
<https://scholarworks.gvsu.edu/gradprojects/299>

This Project is brought to you for free and open access by the Graduate Research and Creative Practice at ScholarWorks@GVSU. It has been accepted for inclusion in Culminating Experience Projects by an authorized administrator of ScholarWorks@GVSU. For more information, please contact scholarworks@gvsu.edu.

CIS693 – MASTER’S PROJECT

Introduction:

In my master’s project, I used Terraform to create a scalable infrastructure on Amazon Web Services (AWS) for my personal website. Terraform is an open-source infrastructure-as-code (IAC) tool that allows you to create, manage and provision infrastructure resources, such as virtual machines, storage accounts, networks, and more, across multiple cloud providers and on-premises data centers using a declarative configuration language. A scalable infrastructure is important because it enables a system or application to handle increasing amounts of traffic or workload without experiencing performance issues or downtime. It ensures that the system remains responsive, available, and reliable as an organization grows or its user base expands. Amazon Web Services (AWS) is a comprehensive cloud computing platform offered by Amazon. It provides a broad range of cloud-based services, including compute, storage, networking, database, analytics, machine learning, security and more. These services are designed to help organizations of all size to build, deploy and scale applications and infrastructure in the cloud with ease and flexibility. This project highlights the benefits of Infrastructure as Code and the power of Terraform with AWS for automating infrastructure deployment.

Background:

For my personal website, I've leveraged a variety of Amazon Web Services (AWS) to ensure optimal performance, security, and reliability. Here are the key AWS tools that I've used and how they contribute to the functionality of my website:

Elastic Cloud Compute (EC2): I've utilized EC2 as my web server to handle all computational tasks required for my website.

Virtual Private Cloud (VPC): To manage traffic and ensure security, I've implemented VPC to regulate access to my website.

Simple Storage Service (S3): All the files needed for my website have been uploaded to S3 buckets, which helps ensure that my site is always accessible and running smoothly.

Route53: This service allowed me to register a domain for my personal website, making it easier for visitors to find and access.

Elastic Load Balancer (ELB): To manage traffic and optimize performance, I've integrated ELB with my web servers, effectively distributing traffic across multiple instances.

Terraform: As my preferred infrastructure-as-code tool, I've utilized Terraform to deploy all of the aforementioned AWS services, streamlining the process and ensuring consistency across environments.

Methodology:

To create the necessary cloud infrastructure to host my personal website, I followed a specific methodology that involved utilizing Terraform to streamline the deployment process. Here are the steps I took:

First, I created an AWS account and generated an access key for authentication purposes. I then used this key to connect Terraform with my AWS account.

To validate my Terraform and AWS configurations were working properly, I created an EC2 instance using Terraform.

I created four separate modules - compute, networking, storage, and database - to handle different aspects of the infrastructure. Within each module, I defined various AWS services and settings using Terraform.

I started by deploying the networking module, which defined subnets, route tables, and other necessary networking components. Once this was set up, I used the VPC ID from the networking module to deploy other services such as autoscaling and elastic load balancing. I then created a DNS record for my website using Route53 and linked it to my load balancer. All the files needed for my website were uploaded to S3. Finally, I created a "main.tf" file that called all the separate modules I had created. This allowed me to deploy the entire infrastructure in a single Terraform deployment.

To ensure security, I defined all sensitive information in a "dev.tfvars" file and referenced it during deployment.

The deployment process involved running the following commands:

```
terraform init
```

```
terraform plan -var-file dev.tfvars
```

```
terraform apply -var-file dev.tfvars
```

By following this methodology, I was able to efficiently and securely deploy all the cloud services needed to host my personal website.

Outcomes:

Instances | EC2 Management

aws.amazon.com/ec2/home?region=us-east-1#instances:instanceState=running_running

instances (4) Info

Find instance by attribute or tag (case-sensitive)

instance state = running | instance state = running | Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
-	i-082b643fcb3fd8d38	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	-
-	i-0e4621fa81ca0039	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	-
masters-srikar	i-0e3462378e4064992	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-52-87-179-1
-	i-04f4381294b47b008	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	-

elect an instance

Details Page | EC2 Management

onsale.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancerDetails:clbName=masters-elb

EC2 | Load balancers | masters-elb

Load balancer: masters-elb

Description | Instances | Health check | Listeners | Monitoring | Tags | Migration

Basic Configuration

Name	masters-elb	Creation time	April 10, 2023 at 1:52:19 AM UTC-4
DNS name	masters-elb-905418596.us-east-1.elb.amazonaws.com (A Record)	Hosted zone	Z35SXDOTRQ7X7K
Type	Classic (Migrate Now)	Status	1 of 3 instances in service
Scheme	internet-facing	VPC	vpc-0eb1e240d967eaf1f
Availability Zones	subnet-0842f4aaf1c3cd9dd - us-east-1b, subnet-07796fea5d0105d5 - us-east-1a		

Port Configuration

Port Configuration	80 (HTTP) forwarding to 80 (HTTP)
	Stickiness: Disabled
	Edit stickiness
	443 (HTTPS, ACM Certificate: f8211102-34c7-4064-bd24-227eedd90548) forwarding to 80 (HTTP)
	Stickiness: Disabled
	Edit stickiness

Security

Source Security Group	sg-00a82706cafe83caa, test-autoscale2
	<ul style="list-style-type: none"> Allow TLS inbound traffic
	sg-035530f812aedcb79, default
	<ul style="list-style-type: none"> default VPC security group
	Edit security groups

Attributes

Idle timeout	400 seconds
--------------	-------------

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy

22 Manager x VPC Management Console x +

s-east-1.console.aws.amazon.com/vpc/home?region=us-east-1#VpcDetails:VpcId=vpc-0eb1e240d967eaf1f

Search [Option+S]

vpc-0eb1e240d967eaf1f / HA-vpc-Masters-Sri

Details Info

VPC ID vpc-0eb1e240d967eaf1f	State Available	DNS hostnames Enabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-0d8d5277bc3c4abba	Main route table rtb-0aa35de9c52b6599c / HA-vpc-Masters-Sri-default	Main network ACL acl-01ca9a7b22a51e1e / HA-vpc-Masters-Sri-default
Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool -	IPv6 CIDR (Network)
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups -	Owner ID 150468161878	

Resource map New CIDRs Flow logs Tags

Resource map Info

VPC Show details
Your AWS virtual network

HA-vpc-Masters-Sri

Subnets (6)
Subnets within this VPC

us-east-1a

- HA-vpc-Masters-Sri-public-us-east-1a
- HA-vpc-Masters-Sri-private-us-east-1a
- HA-vpc-Masters-Sri-db-us-east-1a

us-east-1b

- HA-vpc-Masters-Sri-public-us-east-1b
- HA-vpc-Masters-Sri-db-us-east-1b
- HA-vpc-Masters-Sri-private-us-east-1b

Route tables (4)
Route network traffic to resources

- HA-vpc-Masters-Sri-private-us-east-1a
- HA-vpc-Masters-Sri-default
- HA-vpc-Masters-Sri-public
- HA-vpc-Masters-Sri-private-us-east-1b

back Language © 2023, Amazon Web Services, Inc. or its affiliates

Route 53 Management Console x +

s-east-1.console.aws.amazon.com/route53/home#DomainDetail:srikarpratapmasters.com

Search [Option+S]

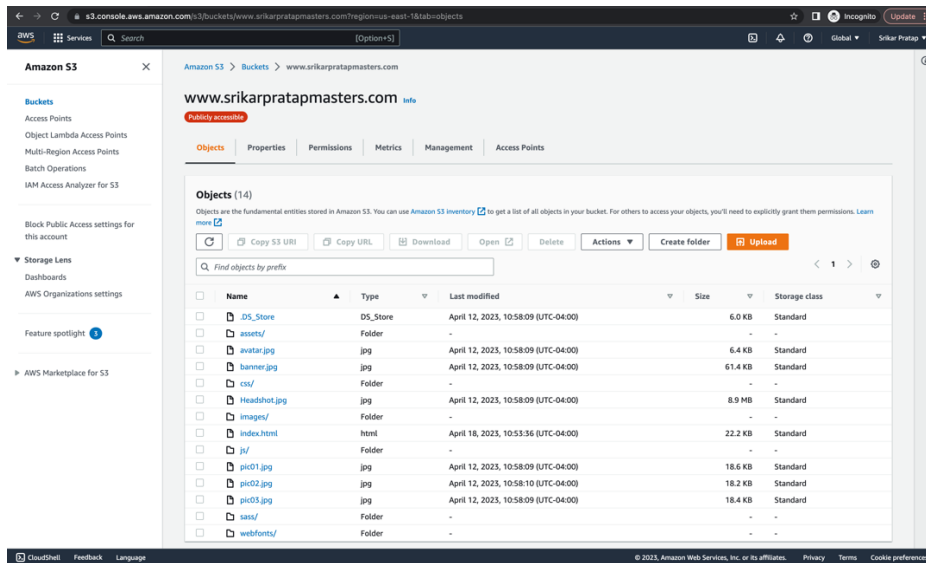
Registered domains > srikarpratapmasters.com

Edit contacts Manage DNS Delete domain

Domain	srikarpratapmasters.com	Transfer lock	Disabled (enable)	Name servers	ns-331.a ns-1166. ns-560.a ns-2012. Act or e
Registration date	2023-04-05	Authorization code	Get code	Domain name status code	addPeriod ok
Expiration date	2024-04-05 (extend)	Tag	View and manage tags for your domains using Tag editor	DNSSEC status	Disabled Manage
Auto renew	Disabled (enable)				

Registrant contact Verified	Administrative contact	Technical contact
Srikar Pratap prataps@mail.gvsu.edu +1.6162062830 4276 Campus View Dr, Allendale MI 49401 US	Srikar Pratap prataps@mail.gvsu.edu +1.6162062830 4276 Campus View Dr, Allendale MI 49401 US	Srikar Pratap prataps@mail.gvsu.edu +1.6162062830 4276 Campus View Dr, Allendale MI 49401 US

Language © 2023, Amazon Web Services, Inc. or its affiliates.



Reflection:

Upon completing this project, I am pleased to report that everything went smoothly, and I was able to gain new skills and knowledge along the way. However, there were a few challenging phases during the project that required me to seek help.

One particular hurdle I encountered was related to hosting my personal website. Fortunately, I was able to reach out to Dr. Erik Fredericks for guidance and support. Thanks to his expertise and advice, I was able to navigate the issue successfully and complete the project as planned.

Overall, I found this project to be a valuable learning experience that helped me expand my understanding of cloud infrastructure and the benefits it can offer. I look forward to applying this newfound knowledge in future projects and endeavors.

Summary:

- The master's project involved using Terraform to create a scalable infrastructure for their personal website on Amazon Web Services (AWS).
- Terraform is an infrastructure-as-code tool that enables the creation and management of infrastructure resources across multiple cloud providers and on-premises data centers. The project highlighted the benefits of Infrastructure as Code and the power of Terraform with AWS for automating infrastructure deployment.
- Created four modules for computing, networking, storage, and database services, and then used a "main.tf" file to call all the modules and deploy the entire infrastructure in a single Terraform deployment.
- Also defined secured information in a "dev.tfvars" file and used Terraform commands such as "init", "plan", and "apply" during deployment.

References:

[1] Terraform Documentation – “<https://developer.hashicorp.com/terraform/docs>”

[2] AWS Documentation – “<https://docs.aws.amazon.com/>”

[3] AWS and Terraform – “<https://developer.hashicorp.com/terraform/tutorials/aws-get-started>”