

**ENPM 665 CLOUD SECURITY**  
**FINAL PROJECT (IMPLEMENTATION OF CLOUD SECURITY**  
**ARCHITECTURE FOR COBRA KAI)**  
**AAKASH RAMAN**  
**UNIVERSITY ID: 119211663**

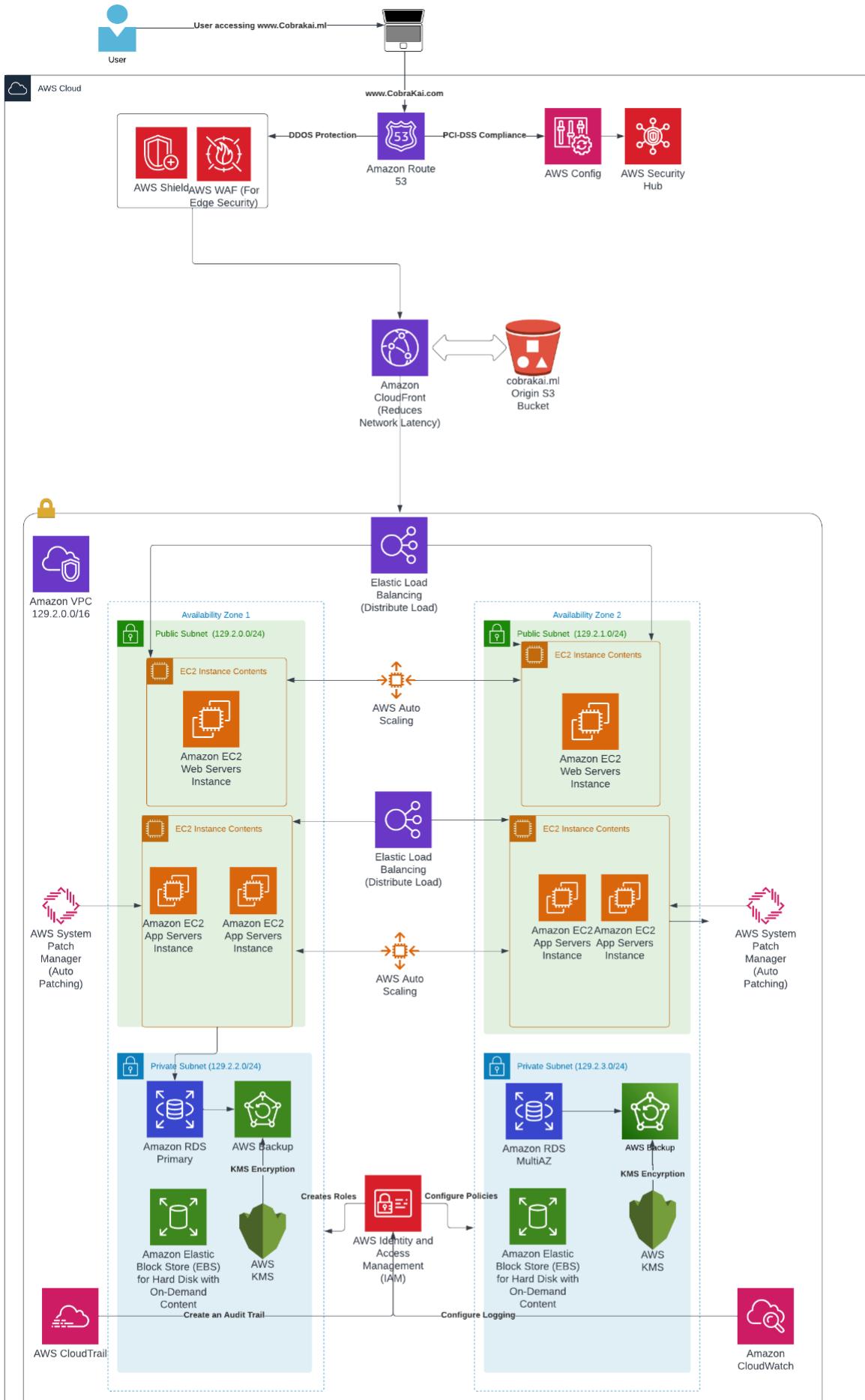
***Introduction***

Now that I have got approval from the upper management to develop the migration of Cobra Kai's services to the Cloud, there are some important recommendations that I would like to highlight in this report.

Some of these include configuring a proper Identity & Access Management Policy for all the employees at Cobra Kai, maintaining regular backups of the Database server, configuring logging and monitoring of services using AWS Cloudwatch & CloudTrail, deploying DoS protection using tools like AWS WAF & AWS Shield, AWS CloudFront and Route 53 for Content Delivery and Web Hosting, ensuring that connections made remotely are in a secure manner instead of being available globally, configuring a System and Patch Manager, and ensuring compliance with PCI-DSS and using Virtual Private Clouds for Networking and managing them using Security Groups and NACLs (Network Access Control Lists).

***Architectural Diagram for Implementation:***

The Architectural Diagram that I am going to implement is shown below.



## Initial Setup of Cloud Environment:

The below screenshots highlight the initial setup of the Cobra Kai application I performed on AWS:

1. The first step is to create the VPC for our Cobra Kai application. This VPC has a CIDR Block of **129.2.0.0/16** and a **Default Tenancy**. It also has a Main Route Table of **rtb-003d9815fae50bd6e** and a Main Network ACL of **acl-079713d8715259850**.

Name	VPC ID	State	IPv4 CIDR	Main route table	Main network ACL	Tenancy
Cobra Kai-VPC-Aakash	vpc-062862b2fab0482fe	Available	129.2.0.0/16	rtb-003d9815fae50bd6e	acl-079713d8715259850	Default

**Details**

VPC ID vpc-062862b2fab0482fe	State <b>Available</b>	DNS hostnames Disabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-032d82a880124a09b	Main route table rtb-003d9815fae50bd6e	Main network ACL acl-079713d8715259850
Default VPC No	IPv4 CIDR 129.2.0.0/16	IPv6 pool -	IPv6 CIDR (Network border group) -
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups -	Owner ID 584895298881	

2. The next step is to create subnets for the VPC we created. Subnetting is the process of dividing the CIDR block into smaller segments. My VPCs CIDR Block is 129.2.0.0/16 which I have divided into 4 subnets and their IDs are shown in the below screenshot. In order to ensure high availability, I have created a Public-Private subnet pair in both “us-east-1a” and “us-east-1b” Availability Zone respectively.

Name	Subnet ID	State	VPC	IPv4 CIDR	Availability Z...	Route table	Network ACL
Private Subnet 2-RDS Replication	subnet-0f41b184b4656676	Available	vpc-062862b2fab0482fe   Cobra Kai-VPC-Aakash	129.2.3.0/24	us-east-1b	rtb-003d9815fae50bd6e	acl-079713d8715259850
Private Subnet 1-Amazon RDS	subnet-00434eea4ba34da30	Available	vpc-062862b2fab0482fe   Cobra Kai-VPC-Aakash	129.2.2.0/24	us-east-1a	rtb-003d9815fae50bd6e	acl-079713d8715259850
Public Subnet 2-CobraKai	subnet-0218816640a0e9bbd	Available	vpc-062862b2fab0482fe   Cobra Kai-VPC-Aakash	129.2.1.0/24	us-east-1b	rtb-003d9815fae50bd6e	acl-079713d8715259850
Public Subnet 1-CobraKai	subnet-0b22395350659338	Available	vpc-062862b2fab0482fe   Cobra Kai-VPC-Aakash	129.2.0.0/24	us-east-1a	rtb-003d9815fae50bd6e	acl-079713d8715259850

3. The next step is to set up 4 EC2 Instances (2 App Servers & 2 Web Servers) of Cobra Kai and configure them with the initial VPC and Subnet I created. I also designed Auto Scaling groups for each of the instances that enhance scalability depending on the load. One Web-App Server is in Public Subnet 1 & the other Web-App Server is in Public Subnet 2. We also need to set up the storage for each of our instances so we can back them up easily.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	IPv4 Public IP	VPC ID	Subnet ID
App Server 1-CobraKai	i-0e1db8f0d6c97e998	t2.micro	us-east-1a	running	2/2 checks ...	44.201.60.138	vpc-062862b2fab0482fe	subnet-0b223953506650338
App Server 2-CobraKai	i-0558cf7f0eb5e6da	t2.micro	us-east-1b	running	2/2 checks ...	3.84.162.25	vpc-062862b2fab0482fe	subnet-0218816640a0e9bbd
Web Server 1-CobraKai	i-09e4b6e6f58117e6f	t2.micro	us-east-1a	running	2/2 checks ...	44.197.219.173	vpc-062862b2fab0482fe	subnet-0b2239535066659338
Web Server 2-CobraKai	i-0f38e2ce5ec0a8f17	t2.micro	us-east-1b	running	2/2 checks ...	52.86.55.87	vpc-062862b2fab0482fe	subnet-0218816640a0e9bbd

The following screenshot shows the EBS (Elastic Block Storage) volumes that I have attached to each of my server's EC2 Instances (Web & App Servers).

Name	Volume ID	Type	Size	Snapshot	Availability Zone	Volume state	Attached Instances	Volume state
Web Server 2-CobraKai	vol-09e1b500626364d87	gp2	8 GiB	snap-0cc18315b85966d60	us-east-1b	In-use	i-0f38e2ce5ec0a8f17 (Web Server 2-CobraKai): /dev/xvda (attached)	Okay
App Server-2 CobraKai	vol-0097cd5be4fb07cd8	gp2	8 GiB	snap-0cc18315b85966d60	us-east-1b	In-use	i-0558cf7f0eb5e6da (App Server 2-CobraKai): /dev/xvda (attached)	Okay
Web Server 1-CobraKai	vol-0a12e625a0489dd5f	gp2	8 GiB	snap-0cc18315b85966d60	us-east-1a	In-use	i-09e4b6e6f58117e6f (Web Server 1-CobraKai): /dev/xvda (attached)	Okay
App Server-1 CobraKai	vol-0e89b97be5572a268	gp2	8 GiB	snap-0cc18315b85966d60	us-east-1a	In-use	i-0e1db8f0d6c97e998 (App Server 1-CobraKai): /dev/xvda (attached)	Okay

4. To achieve automatic scalability for Cobra Kai virtual instances, we can use the concept of Auto Scaling Groups. I have set up 1 Auto Scaling Group “WebServer Cobra-Kai ASG-Aakash” with the desired capacity of 1 instance. This means that 1 EC2 Instance will be created or terminated depending on the load.

**Auto Scaling groups (1/1) Info**

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
WebServer Cobra Kai ASG-Aakash	CobraKai-AutoScalingGrp-Aakash-LT   Version Default	1	-	1	1	1	us-east-1a, us-east-1b

**Auto Scaling group: WebServer Cobra Kai ASG-Aakash**

**Group details**

Desired capacity	1	Auto Scaling group name	WebServer Cobra Kai ASG-Aakash
Minimum capacity	1	Date created	Fri Dec 02 2022 19:04:04 GMT-0500 (Eastern Standard Time)
Maximum capacity	1	Amazon Resource Name (ARN)	arn:aws:autoscaling:us-east-1:584895298881:autoScalingGroup:b225637e-795f-4dc4-90f1-8e8652dbc1:a:autoScalingGroupName/WebServer Cobra Kai ASG-Aakash
Desired capacity type	Units		

- Similarly, I have set up 1 more Auto Scaling Group “AppServer Cobra-Kai ASG-Aakash” with the desired capacity of 1 instance. This means that 1 EC2 Instance will be created or terminated depending on the load.

**Auto Scaling groups (1/2) Info**

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
AppServer Cobra-Kai ASG-Aakash	CobraKai-AutoScalingGrp-Aakash-LT   Version Default	1	-	1	1	1	us-east-1a, us-east-1b

**Auto Scaling group: AppServer Cobra-Kai ASG-Aakash**

**Group details**

Desired capacity	1	Auto Scaling group name	AppServer Cobra-Kai ASG-Aakash
Minimum capacity	1	Date created	Fri Dec 02 2022 19:16:18 GMT-0500 (Eastern Standard Time)
Maximum capacity	1	Amazon Resource Name (ARN)	arn:aws:autoscaling:us-east-1:584895298881:autoScalingGroup:4e9353e3-03d9-4050-868b-6dd60d533df1:a:autoScalingGroupName/AppServer Cobra-Kai ASG-Aakash
Desired capacity type	Units		

- To handle any load, I have created 2 Load Balancers to distribute the load among the EC2 Instances which are added to its Target Group. Here, ALB-1 is Load Balancer 1 which

has the Target Group TG-1 attached to it. Similarly, Load Balancer 2 has Target Group TG-2. The Load Balancer will check the health of the instances and check their status by redirecting the requests to the Registered Targets.

The screenshot shows the AWS EC2 Load Balancers console. The top navigation bar includes 'EC2 > Load balancers > CobraKai-ALB-1'. The main title is 'CobraKai-ALB-1'. On the right, there are 'Actions' and a refresh button. Below the title is a 'Details' section with the ARN: arn:aws:elasticloadbalancing:us-east-1:584895298881:loadbalancer/app/CobraKai-ALB-1/2a8a0e978677e7ae. The table contains the following data:

Load balancer type Application Load Balancer	DNS name CobraKai-ALB-1-680610301.us-east-1.elb.amazonaws.com (A Record)	Status Active	VPC vpc-062862b2fab0482fe
IP address type IPv4	Scheme Internet-facing	Availability Zones subnet-0b223953506659338 us-east-1a (use1-az1) subnet-0218816640a0e9bbd us-east-1b (use1-az2)	Hosted Zone Z355XD0TRQ7X7K

The screenshot shows the AWS EC2 Target Groups console. The top navigation bar includes 'EC2 > Target groups'. The main title is 'Target groups (1/2)'. On the right, there are 'Actions' and a 'Create target group' button. Below the title is a search bar and a table:

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
<input checked="" type="checkbox"/> CobraKai-ALB-1-TG	arn:aws:elasticloadbalancin...	80	HTTP	Instance	CobraKai-ALB-1	vpc-062862b2fab0482fe
<input type="checkbox"/> CobraKai-ALB-2-TG	arn:aws:elasticloadbalancin...	80	HTTP	Instance	CobraKai-ALB-2	vpc-062862b2fab0482fe

Below the table, a modal window titled 'Target group: CobraKai-ALB-1-TG' is open. It shows the following details:

Details			
Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-062862b2fab0482fe
IP address type IPv4	Load balancer CobraKai-ALB-1	Target access analysis <span style="color: blue;"> ⓘ Access issue detected</span>	

The below screenshot shows the Target Group of **CobraKai-ALB-1** where our Web Servers are added as **Registered Targets** and are “**healthy**”.

The load balancer starts routing requests to a newly registered target as soon as the registration process completes and the target passes the initial health checks. If demand on your targets increases, you can register additional targets. If demand on your targets decreases, you can deregister targets.

**Edit**

Instance ID	Name	Port	Availability Zone	Status	Description
i-09e4b6e6f58117e6f	Web Server 1-CobraKai	80	us-east-1a	healthy	This target is currently passing target group's health checks.
i-0f38e2ce5ec0a8f17	Web Server 2-CobraKai	80	us-east-1b	healthy	This target is currently passing target group's health checks.

The below screenshot shows the Target Group of **CobraKai-ALB-2** where our App Servers are added as **Registered Targets** and are “**healthy**”.

The load balancer starts routing requests to a newly registered target as soon as the registration process completes and the target passes the initial health checks. If demand on your targets increases, you can register additional targets. If demand on your targets decreases, you can deregister targets.

**Edit**

**Target group:** CobraKai-ALB-2-TG

Instance ID	Name	Port	Availability Zone	Status	Description
i-0e1db8f0d6c97e998	App Server 1-CobraKai	80	us-east-1a	healthy	This target is currently passing target group's health checks.
i-0558cfdf70eb5e6da	App Server 2-CobraKai	80	us-east-1b	healthy	This target is currently passing target group's health checks.

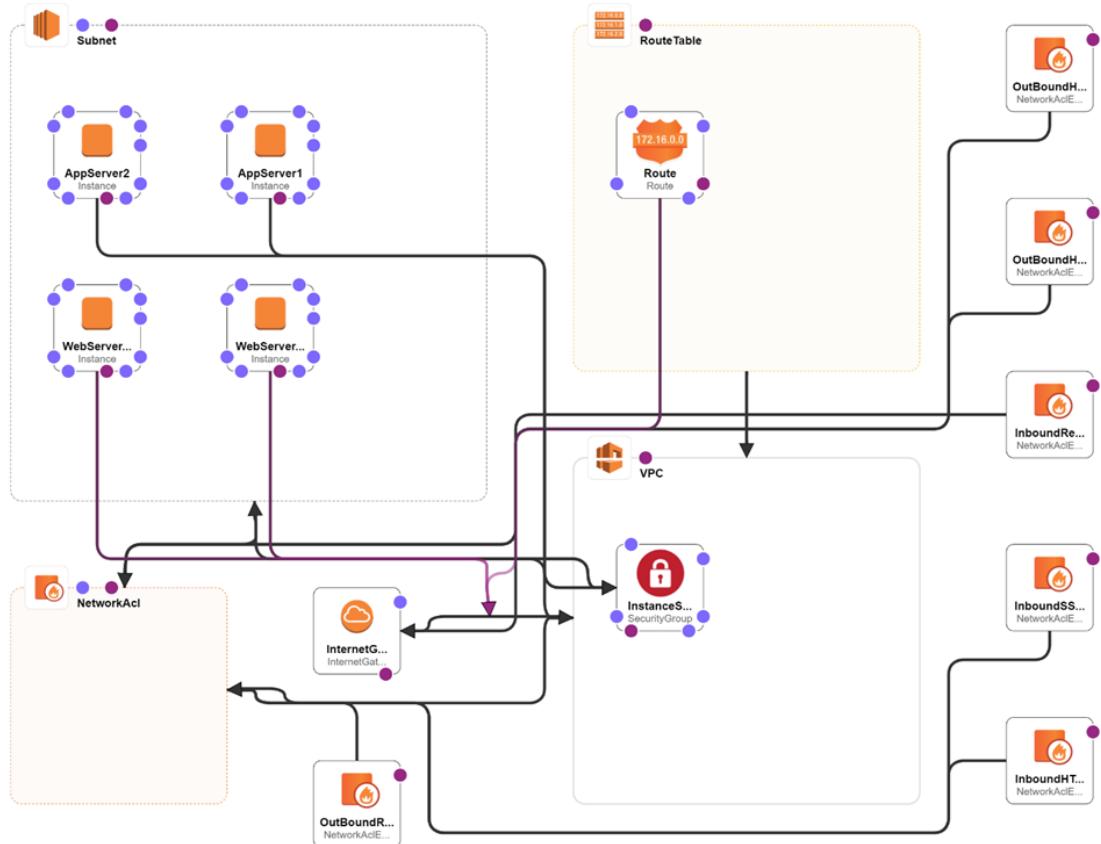
## ***Recommendations for Automatic Patch Management using AWS System Manager:***

Patching resources is always a solid Security practice that is very useful to reduce the attack surface and risk of vulnerabilities affecting Cobra Kai's assets. AWS uses the concept of AWS SSM to achieve Patch Management.

AWS SSM (System Manager) is a platform that provides automated patching capabilities, Fleet, Node & Change Management for all the instances launched that have in-built security controls. AWS SSM can provide automated patching for both Windows & Linux-based instances.

I have used the concepts of **Infrastructure as Code** and **Operations as Code** to first set up an AWS SSM template from this [link](#) from AWS Labs and then used AWS CloudFormation to deploy that template on my instances to set up the patching.

1. The below screenshot highlight the architecture of the AWS System Manager, which I performed on AWS. You can see it in the “**View in Designer**” option while creating a CloudFormation Stack.



2. AWS SSM requires an **IAM Role** in order to allow users to execute commands and instances to process those commands. Hence, the below screenshot depicts a “**ManagedInstancesRole**” that allows EC2 Instances to call AWS services based on the user's needs.

The screenshot shows the AWS IAM Role details page for a role named "ManagedInstancesRole-Aakash". The page includes fields for Role name, Description, and a JSON policy editor. The policy editor displays the following JSON code:

```

1  [
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": [
7                  "sts:AssumeRole"
8              ],
9              "Principal": {
10                 "Service": [
11                     "ec2.amazonaws.com"
12                 ]
13             }
14         }
15     ]
16 ]

```

3. The below screenshot depicts the policies that I have added to the above Role. Here, the “**AmazonSSMManagedInstanceCore**” is used.

The screenshot shows the AWS IAM Roles page. The top navigation bar includes 'Global' and 'Aakash @ 5848-9529-8881'. The main content displays the 'ManagedInstancesRole-Aakash' role, which allows EC2 instances to call AWS services on your behalf. The 'Summary' section provides details like creation date (December 03, 2022), last activity (18 minutes ago), ARN (arn:aws:iam::584895298881:role/ManagedInstancesRole-Aakash), and maximum session duration (1 hour). Below the summary, tabs for 'Permissions', 'Trust relationships', 'Tags', 'Access Advisor', and 'Revoke sessions' are visible. The 'Permissions' tab is selected, showing one managed policy attached: 'AmazonSSMManagedInstanceCore'. This policy is described as 'The policy for Amazon EC2 Role to ena...'.

4. The below screenshots show the “**AmazonSSMManagedInstanceCore**” policy in JSON Format.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeAssociation",
                "ssm:GetDeployablePatchSnapshotForInstance",
                "ssm:GetDocument",
                "ssm:DescribeDocument",
                "ssm:GetManifest",
                "ssm:GetParameter",
                "ssm:GetParameters",
                "ssm>ListAssociations",
                "ssm>ListInstanceAssociations",
                "ssm:PutInventory",
                "ssm:PutComplianceItems",
                "ssm:PutConfigurePackageResult",
                "ssm:UpdateAssociationStatus",
                "ssm:UpdateInstanceAssociationStatus",
                "ssm:UpdateInstanceInformation"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

```

        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "ec2messages:AcknowledgeMessage",
            "ec2messages:DeleteMessage",
            "ec2messages:FailMessage",
            "ec2messages:GetEndpoint",
            "ec2messages:GetMessages",
            "ec2messages:SendReply"
        ],
        "Resource": "*"
    }
]
}

```

- Now that a Role has been attached to AWS SSM, the next step is to ensure that all the instances have an **SSM Agent installed**. This is crucial for the Patching process to take place. Luckily, as a new feature, AWS already installs SSM Agents for each instance according to the Operating System.

Once that is done, all the Web and App servers are added to AWS FleetManager and their status is “**Online**” so that they are ready to get patched as shown by the below screenshot:

	Node ID	Node state	Node name	Platform	Operating System	Source type	SSM Agent ping s...	SSM Agent ve...	Image ID	EC2 instance
<input type="checkbox"/>	i-0558cf7f0eb5e6da	Running	App Server 2-CobraKai	Linux	Amazon Linux	EC2 instance	Online	3.1.1732.0	ami-0b0db5067f052a63	<input checked="" type="checkbox"/>
<input type="checkbox"/>	i-0e1db8f0d6c97e998	Running	App Server 1-CobraKai	Linux	Amazon Linux	EC2 instance	Online	3.1.1732.0	ami-0b0db5067f052a63	<input checked="" type="checkbox"/>
<input type="checkbox"/>	i-0f38e2ce5ec0abf17	Running	Web Server 2-CobraKai	Linux	Amazon Linux	EC2 instance	Online	3.1.1732.0	ami-0b0db5067f052a63	<input checked="" type="checkbox"/>
<input type="checkbox"/>	i-09e4b6e6f58117e6f	Running	Web Server 1-CobraKai	Linux	Amazon Linux	EC2 instance	Online	3.1.1732.0	ami-0b0db5067f052a63	<input checked="" type="checkbox"/>

- Next, a **Patch Baseline** needs to be established depending on the Operating System that defines what patches need to be installed, the rules for the Patching process as well as the timeline for patching. This way CobraKai can choose maintenance hours for patching their servers and reduce their attack surface.

7. AWS also has some predefined Patch Baselines and we can use those Baselines for Patching as well. In the below screenshot, I am using the “SSM run command” to perform the Patching process using the predefined **AWS-RunPatchBaseline**. This is used for immediate Patching rather than having a Scheduled Patching.

8. The AWS SSM also produces a **Patch Management Compliance Report** which judges how well the instances that are scheduled to be patched are Patch Compliant or have any configuration inconsistencies. This is shown in the below screenshot.

Hence, Cobra Kai can use the AWS SSM to regularly patch all their servers and ensure that updates are performed in a timely manner.

## ***Recommendation for Handling DDoS Attacks & Alternative Routes using Route 53 & AWS WAF:***

The main idea to handle DDoS attacks and create a secure alternative route to the website, “[www.cobrakai.ml](http://www.cobrakai.ml)” is to route traffic using Route 53 and then set up AWS WAF/Shield to protect that incoming traffic and enhance resiliency and reduce Network Latency, use a Content Distribution Network, AWS CloudFront.

1. I am planning to host the Cobra Kai Website using a Free Domain Registrar “www.Freenom.com”. The URL I used was “www.cobrakai.ml”. The process to set up AWS Static Web Hosting along with the Sample Website was given in <https://github.com/kts262/enpm809j>.
  
- A. The below screenshot shows all the DNS Records (A, SOA & NS) of “[www.cobrakai.ml](http://www.cobrakai.ml)” of the hosted zone I created in Route 53:

The screenshot displays the AWS Route 53 Hosted Zones interface. At the top, it shows the navigation path: Route 53 > Hosted zones > cobrakai.ml. The main section is titled "Hosted zone details" for the zone "Public cobrakai.ml". It lists the following information:

Hosted zone name	Query log	Name servers
Hosted zone name: cobrakai.ml	Query log: -	Name servers: ns-2003.awsdns-58.co.uk ns-1060.awsdns-04.org ns-268.awsdns-33.com ns-720.awsdns-26.net
Hosted zone ID	Type	
Z05885231CXNUUXJ2Q8EH	Public hosted zone	
Description	Record count	
-	4	

Below this, there are tabs for "Records (4)", "DNSSEC signing", and "Hosted zone tags (0)". The "Records (4)" tab is selected, showing a table of DNS records:

Record name	Type	Routing policy	Value/Route traffic to
www.cobrakai.ml	A	Simple	s3-website-us-east-1.amazonaws.com. ns-2003.awsdns-58.co.uk. ns-1060.awsdns-04.org. ns-268.awsdns-33.com. ns-720.awsdns-26.net.
cobrakai.ml	NS	Simple	ns-2003.awsdns-58.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
cobrakai.ml	SOA	Simple	ns-2003.awsdns-58.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
www.cobrakai.ml	A	Simple	3.227.20.75

- B. Then I added the Name Servers that AWS provided in Freenom.com's Name Servers as shown below.

The screenshot shows the 'Managing cobrakai.ml' page in the Freenom DNS management tools. At the top, there are navigation links: Information, Upgrade, Management Tools (with a dropdown arrow), and Manage Freenom DNS. A blue banner at the top indicates 'Changes Saved Successfully!'. Below this, the 'Nameservers' section is titled 'Nameservers'. It contains a note: 'You can change where your domain points to here. Please be aware changes can take up to 24 hours to propagate.' Two radio button options are present: 'Use default nameservers (Freenom Nameservers)' (unchecked) and 'Use custom nameservers (enter below)' (checked). Below these options, five name server entries are listed, each consisting of a label and a text input field:

- Nameserver 1: NS-1060.AWSDNS-04.ORG
- Nameserver 2: NS-2003.AWSDNS-58.CO.UK
- Nameserver 3: NS-268.AWSDNS-33.COM
- Nameserver 4: NS-720.AWSDNS-26.NET
- Nameserver 5: (empty)

A blue progress bar is visible at the bottom of the form.

- C. Now we have to create an S3 bucket to statically host “[www.cobrakai.ml](http://www.cobrakai.ml)”. Along with NS Records, A Records also need to be added to route traffic correctly via Route 53.

**cobrakai.ml** Info

Publicly accessible

Objects Properties Permissions Metrics Management Access Points

**Objects (6)**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[C](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	about.html	html	December 4, 2022, 15:44:05 (UTC-05:00)	1.3 KB	Standard
<input type="checkbox"/>	buy.html	html	December 4, 2022, 15:44:05 (UTC-05:00)	484.0 B	Standard
<input type="checkbox"/>	index.html	html	December 4, 2022, 15:52:57 (UTC-05:00)	772.0 B	Standard
<input type="checkbox"/>	leadership.html	html	December 4, 2022, 15:44:04 (UTC-05:00)	1.3 KB	Standard
<input type="checkbox"/>	static/	Folder	-	-	-
<input type="checkbox"/>	videos.html	html	December 4, 2022, 15:44:05 (UTC-05:00)	1002.0 B	Standard

- D. This is done by adding Web Server 1's Public IP Address as a Value in the A Record so that traffic will be redirected there (refer to point A). This is depicted by the below screenshot:

Instances (1/1) Info

[C](#) [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find instance by attribute or tag (case-sensitive)

[Instance state = running](#) [Clear filters](#)

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instanc...	Status check	Availa...	Public IPv4 ...	VPC ID
<input checked="" type="checkbox"/>	Web Server 1-CobraKai	i-09e4b6e6f58117e6f	<a href="#">Running</a>	<a href="#">t2.micro</a>	<a href="#">2/2 checks passed</a>	us-east-1a	3.227.20.75	vpc-0628

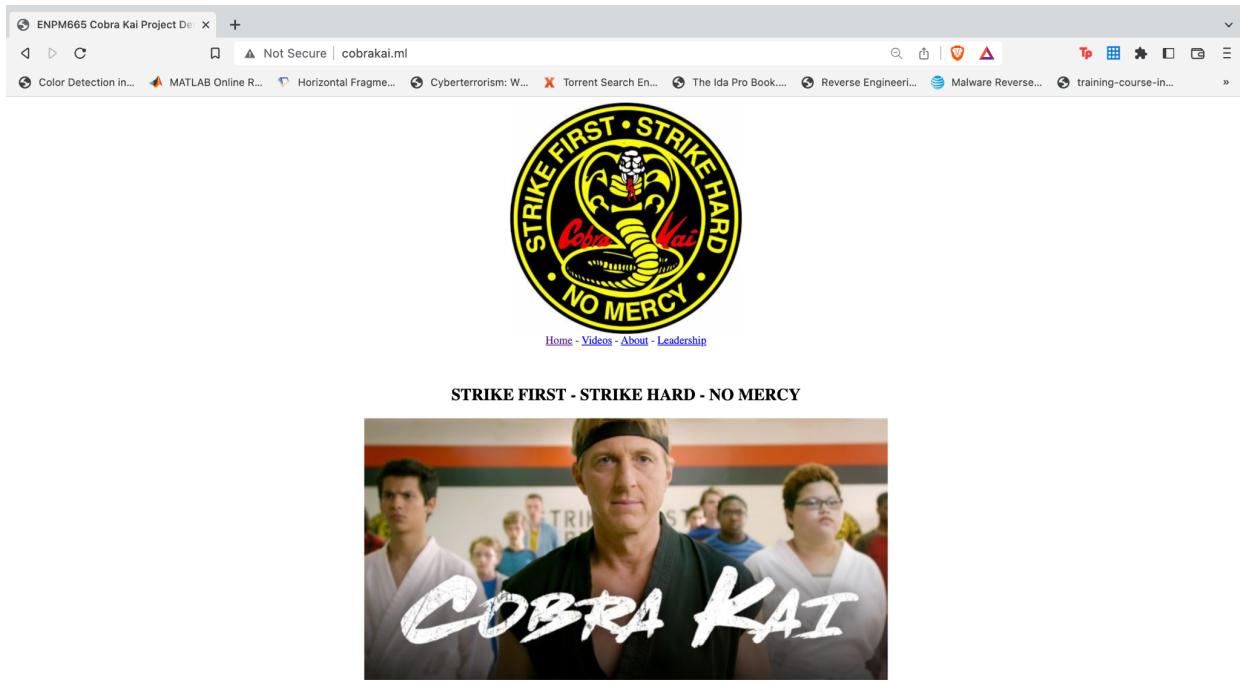
**Instance: i-09e4b6e6f58117e6f (Web Server 1-CobraKai)**

[Details](#) [Security](#) [Networking](#) [Storage](#) [Status checks](#) [Monitoring](#) [Tags](#)

**Instance summary** Info

Instance ID <a href="#">i-09e4b6e6f58117e6f (Web Server 1-CobraKai)</a>	Public IPv4 address <a href="#">3.227.20.75   open address</a>	Private IPv4 addresses <a href="#">129.2.0.190</a>
IPv6 address -	Instance state <a href="#">Running</a>	Public IPv4 DNS -
Hostname type Resource name: i-09e4b6e6f58117e6f.ec2.internal	Private IP DNS name (IPv4 only) <a href="#">ip-129-2-0-190.ec2.internal</a>	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding <a href="#">Opt-in to AWS Compute Optimizer for recommendation s.</a>
Auto-assigned IP address <a href="#">3.227.20.75 [Public IP]</a>	VPC ID <a href="#">vpc-062862b2fab0482fe (Cobra Kai-VPC-Aakash)</a>	<a href="#">View more</a>

- E. After doing this, I managed to host the Cobra Kai website by my domain "[www.cobrakai.ml](http://www.cobrakai.ml)" as shown in the screenshot below.



This is the official site of Cobra Kai, a karate dojo in Reseda, Los Angeles, California! Please check out our [video](#) collections and purchase as many videos as you think make sense!

Thank you for your patience while we work through these IT issues, I'm not a nerd. -- Sensei Johnny Lawrence

- Now, that the static website is hosted, we need to add AWS Rule Sets for the AWS WAF & Shield (Web Application Firewall) so that all common Web Application Vulnerabilities, such as SQL Injection, Cross-Site-Scripting, CSRF, etc are checked for and prevented. The AWS Rule Sets I have added are shown in the below screenshot:

Name	Action	Priority	Custom response
AWS-AWSManagedRulesAmazonIpReputationList	Use rule actions	0	-
AWS-AWSManagedRulesKnownBadInputsRuleSet	Use rule actions	1	-
AWS-AWSManagedRulesSQLRuleSet	Use rule actions	2	-
AWS-AWSManagedRulesLinuxRuleSet	Use rule actions	4	-
AWS-AWSManagedRulesCommonRuleSet	Use rule actions	7	-
AWS-AWSManagedRulesAnonymousIpList	Use rule actions	8	-
AWS-AWSManagedRulesPHPRuleSet	Use rule actions	9	-

3. The below screenshot shows the associated AWS Resources for the WAF. There are mainly 4 types of resources under AWS WAF, namely, Amazon API Gateway, AWS AppSync, Application Load Balancer, and Amazon Cognito User Pools. In this case, I have chosen our Internet-facing Load Balancers of Cobra Kai.

The screenshot shows the AWS WAF console. The navigation path is AWS WAF > Web ACLs > Cobra\_Kai\_Web\_Application\_Firewall. The main tab 'Associated AWS resources' is selected. A search bar at the top says 'Find associated AWS resources'. Below it is a table with two rows:

Name	Resource type	Region
CobraKai-ALB-1	Application Load Balancer	US East (N. Virginia)
CobraKai-ALB-2	Application Load Balancer	US East (N. Virginia)

Buttons for 'Disassociate' and 'Add AWS resources' are visible above the table. A 'Download web ACL as JSON' button is located in the top right corner.

This way all the traffic entering Cobra Kai will be routed via Route 53 and will have to pass AWS WAF, hence reducing the chances of a DDoS attack.

### ***Recommendation for Handling PCI-DSS Compliance using AWS Security Hub:***

Since the clients' credit card information is exposed to attacks, the current Cobra Kai infrastructure does not meet PCI-DSS compliance criteria, which is a major problem.

One tool that performs Security Best Practices checks, automated retrieval, and aggregation of logs is AWS Security Hub.

I have set up AWS Security Hub to monitor whether Cobra Kai is compliant with PCI-DSS. To set up AWS Security Hub you need to, enable AWS Config and then set up AWS Security Hub.

The below screenshot shows the output of the PCI-DSS Report generated by AWS Security Hub. Many of the vulnerabilities are Critical, High, Medium, Low & Informational.

AWS Services Search [Option+5] N. Virginia Aakash Raman

## PCI DSS v3.2.1

### Overview

Security score: -

All enabled	Failed	Unknown	No data	Passed	Disabled
<b>45</b>	0	0	<b>45</b>	0	0

All enabled controls (45) Download

Compliance Status	Severity	ID	Title	Failed checks
○ No Data	■ Critical	PCI.CodeBuild.1	CodeBuild GitHub or Bitbucket source repository URLs should use OAuth	0 of 0
○ No Data	■ Critical	PCI.CodeBuild.2	CodeBuild project environment variables should not contain clear text credentials	0 of 0
○ No Data	■ Critical	PCI.DMS.1	Database Migration Service replication instances should not be public	0 of 0
○ No Data	■ Critical	PCLEC2.1	EBS snapshots should not be publicly restorable	0 of 0
○ No Data	■ Critical	PCLES.1	Elasticsearch domains should be in a VPC	0 of 0
○ No Data	■ Critical	PCI.IAM.1	IAM root user access key should not exist	0 of 0
○ No Data	■ Critical	PCI.IAM.4	Hardware MFA should be enabled for the root user	0 of 0
○ No Data	■ Critical	PCI.IAM.5	Virtual MFA should be enabled for the root user	0 of 0
○ No Data	■ Critical	PCI.Lambda.1	Lambda function policies should prohibit public access	0 of 0
○ No Data	■ Critical	PCI.Opensearch.1	OpenSearch domains should be in a VPC	0 of 0
○ No Data	■ Critical	PCI.RDS.1	RDS snapshot should be private	0 of 0
○ No Data	■ Critical	PCI.RDS.2	RDS DB instances should prohibit public access, as determined by the PubliclyAccessible configuration	0 of 0
○ No Data	■ Critical	PCI.Redshift.1	Amazon Redshift clusters should prohibit public access	0 of 0
○ No Data	■ Critical	PCI.S3.2	S3 buckets should prohibit public read access	0 of 0
○ No Data	■ High	PCI.IAM.3	IAM policies should not allow full *** administrative privileges	0 of 0
○ No Data	■ High	PCI.SSM.1	EC2 instances managed by Systems Manager should have a patch compliance status of COMPLIANT after a patch installation	0 of 0
○ No Data	■ High	PCI.SageMaker.1	Amazon SageMaker notebook instances should not have direct internet access	0 of 0
○ No Data	■ Medium	PCI.CloudTrail.1	CloudTrail should have encryption at-rest enabled	0 of 0
○ No Data	■ Medium	PCI.CloudTrail.3	CloudTrail log file validation should be enabled	0 of 0
○ No Data	■ Medium	PCI.Config.1	AWS Config should be enabled	0 of 0
○ No Data	■ Medium	PCI.EC2.6	VPC flow logging should be enabled in all VPCs	0 of 0
○ No Data	■ Medium	PCI.ELBv2.1	Application Load Balancer should be configured to redirect all HTTP requests to HTTPS	0 of 0
○ No Data	■ Medium	PCI.ES.2	Elasticsearch domains should have encryption at-rest enabled	0 of 0
○ No Data	■ Medium	PCI.IAM.8	Password policies for IAM users should have strong configurations	0 of 0
○ No Data	■ Medium	PCI.IAM.6	MFA should be enabled for all IAM users	0 of 0
○ No Data	■ Medium	PCI.IAM.7	Unused IAM user credentials should be removed	0 of 0
○ No Data	■ Medium	PCI.KMS.1	AWS KMS key rotation should be enabled	0 of 0
○ No Data	■ Medium	PCI.Opensearch.2	OpenSearch domains should have encryption at rest enabled	0 of 0

⌚ No Data	■ Medium	PCI.S3.6	S3 Block Public Access setting should be enabled	0 of 0
⌚ No Data	■ Medium	PCI.S3.4	S3 buckets should have server-side encryption enabled	0 of 0
⌚ No Data	■ Medium	PCI.S3.5	S3 buckets should require requests to use Secure Socket Layer	0 of 0
⌚ No Data	■ Medium	PCI.SSM.3	EC2 instances should be managed by AWS Systems Manager	0 of 0
⌚ No Data	■ Low	PCI.AutoScale.1	Auto scaling groups associated with a load balancer should use load balancer health checks	0 of 0
⌚ No Data	■ Low	PCI.CloudTrail.4	CloudTrail trails should be integrated with Amazon CloudWatch Logs	0 of 0
⌚ No Data	■ Low	PCI.CW.1	A log metric filter and alarm should exist for usage of the "root" user	0 of 0
⌚ No Data	■ Low	PCI.EC2.4	Unused EC2 EIPs should be removed	0 of 0
⌚ No Data	■ Low	PCI.IAM.2	IAM users should not have IAM policies attached	0 of 0
⌚ No Data	■ Low	PCI.Lambda.2	Lambda functions should be in a VPC	0 of 0
⌚ No Data	■ Low	PCI.S3.3	S3 buckets should have cross-region replication enabled	0 of 0
⌚ No Data	■ Low	PCI.SSM.2	EC2 instances managed by Systems Manager should have an association compliance status of COMPLIANT	0 of 0

AWS Security Hub enables us to check for PCI-DSS Compliance by monitoring Cobra Kai's instances and then using the predefined benchmarks to check if they are compliant or not.

The below screenshot shows how I have set Cobra Kai's Web & App Server instances to follow PCI DSS Compliance benchmarks and they have passed them based on AWS Security Hub's Observation.

The screenshot shows the AWS Security Hub Findings page. The title bar includes 'Security Hub' and 'Findings'. Below the title, there is a search bar with the placeholder 'Title starts with PCI' and a 'Add filters' button. The main table displays the following findings:

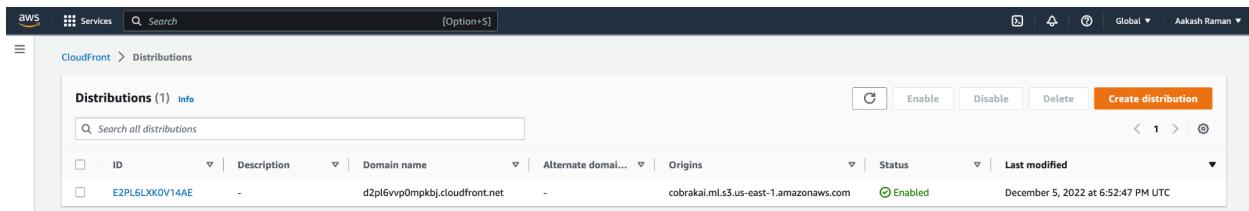
Severity	Workflow status	Record State	Region	Account Id	Company	Product	Title	Resource	Compliance Status	Updated at
■ INFORMATIONAL	RESOLVED	ACTIVE	us-east-1	584895298881	AWS	Security Hub	PCI.EC2.5 Security groups should not allow ingress from 0.0.0.0/0 to port 22	EC2 Security Group default	PASSED	4 minutes ago
■ INFORMATIONAL	RESOLVED	ACTIVE	us-east-1	584895298881	AWS	Security Hub	PCI.S3.4 S3 buckets should have server-side encryption enabled	S3 bucket cf-templates-ppsa3y0wme1v-us-east-1	PASSED	4 minutes ago
■ INFORMATIONAL	RESOLVED	ACTIVE	us-east-1	584895298881	AWS	Security Hub	PCI.S3.1 S3 buckets should prohibit public write access	S3 bucket cobrakai.ml	PASSED	4 minutes ago
■ INFORMATIONAL	RESOLVED	ACTIVE	us-east-1	584895298881	AWS	Security Hub	PCI.S3.1 S3 buckets should prohibit public write access	S3 Bucket cf-templates-ppsa3y0wme1v-us-east-1	PASSED	4 minutes ago
■ INFORMATIONAL	RESOLVED	ACTIVE	us-east-1	584895298881	AWS	Security Hub	PCI.S3.1 S3 buckets should prohibit public write access	S3 Bucket config-bucket-584895298881	PASSED	4 minutes ago
■ INFORMATIONAL	RESOLVED	ACTIVE	us-east-1	584895298881	AWS	Security Hub	PCI.EC2.1 EBS snapshots should not be publicly restorable	Account 584895298881	PASSED	4 minutes ago
■ INFORMATIONAL	RESOLVED	ACTIVE	us-east-1	584895298881	AWS	Security Hub	PCI.IAM.5 Virtual MFA should be enabled for the root user	Account 584895298881	PASSED	4 minutes ago
■ INFORMATIONAL	RESOLVED	ACTIVE	us-east-1	584895298881	AWS	Security	PCI.EC2.4 Unused EC2 EIPs	AwsEc2Eip eipalloc-	PASSED	4 minutes ago

Thus, by incorporating AWS Security Hub into each of Cobra Kai's resources, proper benchmarks can be followed to ensure that PCI-DSS Compliance is maintained regularly.

### ***Recommendations for Handling Resiliency & Reduced Network Latency using Amazon CloudFront:***

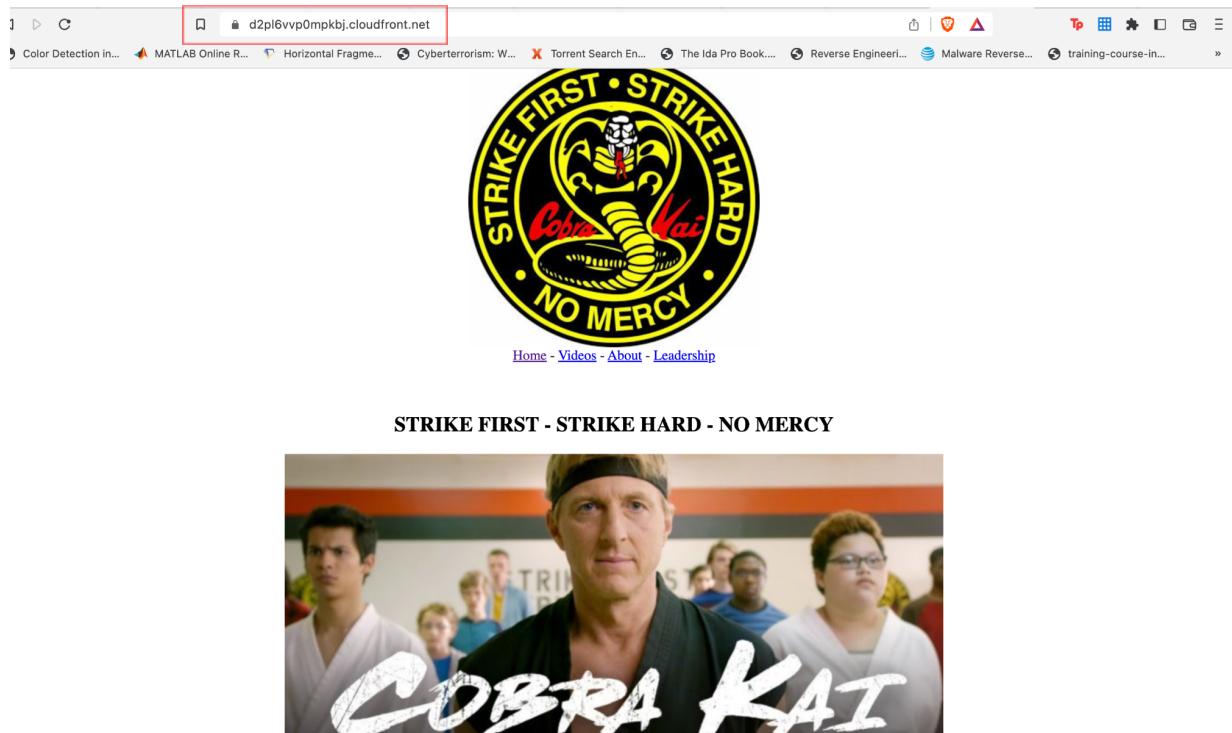
Amazon CloudFront is a Content Delivery Network that provides a global network of proxy servers that cache content which can be videos, audio, images, etc thus reducing the Network Latency and speeding up the delivery of any application to its customers. Logically, incorporating this tool will help Cobra Kai provide its high-demand videos easily and efficiently to its customers.

1. Through Amazon CloudFront, I can access my Cobra Kai website that I deployed in the S3 Bucket securely. The below screenshot depicts the Amazon CloudFront Distribution I used to set up the “www.cobrakai.ml” website.



The screenshot shows the AWS CloudFront Distributions page. At the top, there is a search bar and a dropdown menu labeled "CloudFront > Distributions". Below the search bar, there is a table titled "Distributions (1) Info". The table has columns for ID, Description, Domain name, Alternate domain..., Origins, Status, and Last modified. A single row is visible, showing the ID "E2PL6LXKOV14AE", Domain name "d2p16vvp0mpkbj.cloudfront.net", Origins "cobrakai.ml.s3.us-east-1.amazonaws.com", Status "Enabled", and Last modified "December 5, 2022 at 6:52:47 PM UTC". There are buttons for "Create distribution", "Enable", "Disable", and "Delete".

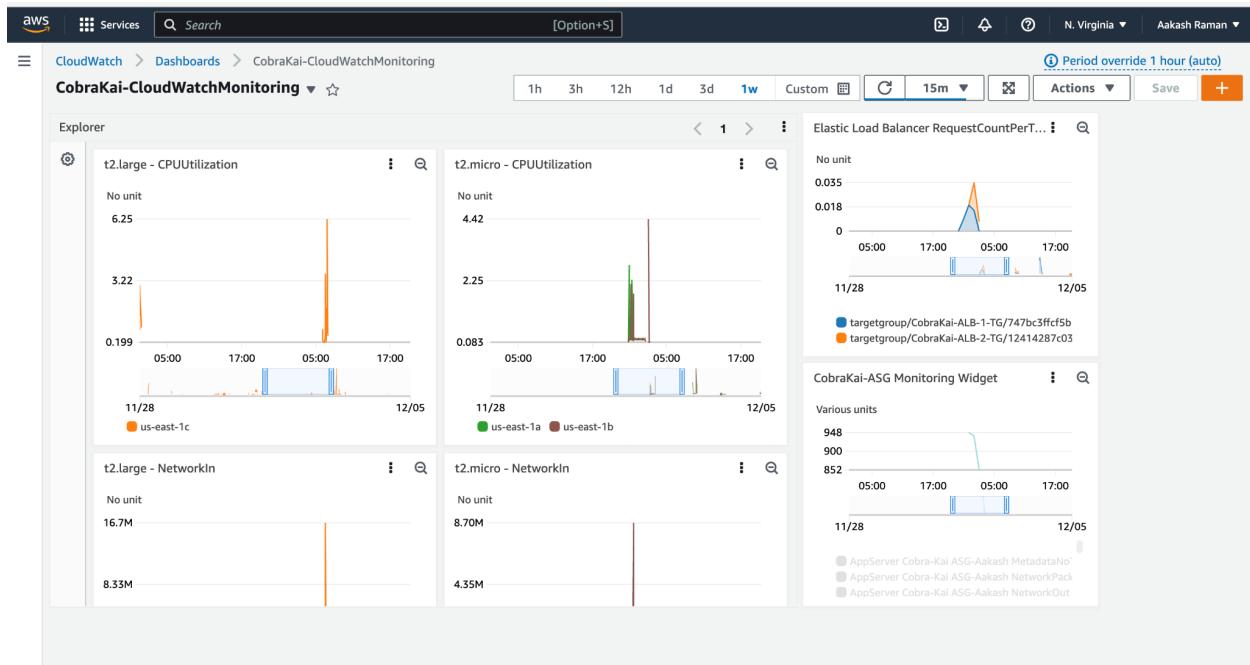
2. By using the domain name from CloudFront distribution, I can load a **secure** “index.html” page of Cobra Kai’s website.



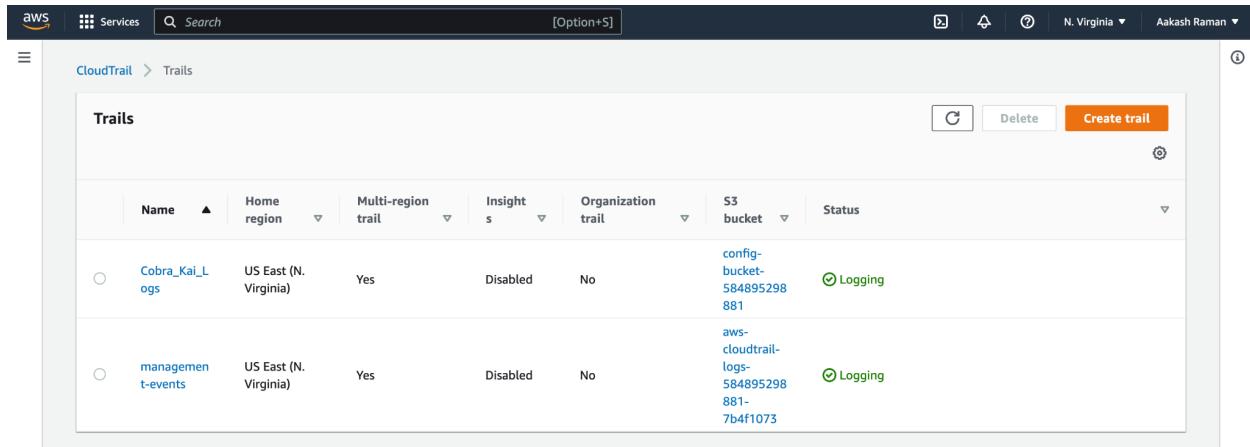
## ***Recommendations for Continuous Logging & Monitoring using Amazon CloudWatch & Amazon CloudTrail:***

Amazon CloudWatch is Amazon's continuous monitoring service that provides valuable insights into how resources that are deployed in the Cloud environment are used regularly. Amazon CloudTrail is a similar tool that handles auditing, troubleshooting, and security monitoring by keeping track of a user's activity along with API usage and calls.

1. To monitor the performance of Cobra Kai's App and Web Servers, I have also configured a Dashboard in Amazon CloudWatch to monitor their network parameters so it can be a centralized location for continuous monitoring of all the critical assets of Cobra Kai.



- Along with the dashboard I created, I have set up logging using AWS CloudTrail Logs that uses KMS (Key Management Service) Encryption where I have created a Customer Managed Key to create a Trail, and the logs are stored in an S3 bucket as shown below in the Trail Log:



- The below screenshot depicts the location of the Cobra\_Kai\_Logs in AWS and it can also be seen that logging is configured and operational for the VPC I set where all the events taken by the users in that VPC get logged. Also, Management & Data Events get logged too.

The screenshot shows the AWS CloudTrail Trails page. At the top, there are navigation links for 'CloudTrail' and 'Trails', followed by the ARN of the trail: 'arn:aws:cloudtrail:us-east-1:584895298881:trail/Cobra\_Kai\_Logs'. On the right side, there are two buttons: 'Delete' and 'Stop logging'. Below these buttons is an 'Edit' button. The main section is titled 'General details' and contains several configuration items:

Setting	Value
Trail logging	Logging
Trail name	Cobra_Kai_Logs
Multi-region trail	Yes
Apply trail to my organization	Not enabled
Trail log location	config-bucket-584895298881/AWSLogs/584895298881
Last log file delivered	-
Log file validation	Enabled
Last file validation delivered	-
SNS notification delivery	Disabled
Log file SSE-KMS encryption	Enabled
AWS KMS key	arn:aws:kms:us-east-1:584895298881:key/b5d157bd-a6a0-40b9-96dd-1c002456c6ab
AWS KMS key alias	CobraKai-Logs-KMS-Key

4. This is the location where the S3 Bucket that we configured for logging is stored.

The screenshot shows the Amazon S3 Buckets page. The URL in the address bar is 'Amazon S3 > Buckets > config-bucket-584895298881 > AWSLogs/ > 584895298881/'. The page title is '584895298881/'. There are two tabs at the top: 'Objects' (which is selected) and 'Properties'. Below the tabs, there is a search bar with the placeholder 'Find objects by prefix' and a 'Upload' button. A row of action buttons includes 'Copy S3 URI', 'Copy URL', 'Download', 'Open', and 'Delete'. The main area displays a table of objects:

	Name	Type	Last modified
<input type="checkbox"/>	CloudTrail-Digest/	Folder	-
<input type="checkbox"/>	CloudTrail/	Folder	-

5. Additionally, as AWS CloudTrail is an auditing tool, all the audits can be seen in the **Event History** section of AWS CloudTrail as shown below:

The screenshot shows the AWS CloudTrail Event History interface. At the top, there's a navigation bar with 'CloudTrail' and 'Event history'. Below it is a header with 'Event history (50+)' and a 'Info' link. A note says 'Event history shows you the last 90 days of management events.' There are buttons for 'Download events' and 'Create Athena table'. A search bar filters by 'AWS::EC2::Instance'. The main area is a table with the following data:

<input type="checkbox"/>	Event name	Event time	User name	Event source	Resource type	R
<input type="checkbox"/>	TerminateInstances	December 03, 2022, 23:11:39 (U...)	Aakash	ec2.amazonaws.com	AWS::EC2::Instance	i-
<input type="checkbox"/>	TerminateInstances	December 03, 2022, 23:11:38 (U...)	Aakash	ec2.amazonaws.com	AWS::EC2::Instance	i-
<input type="checkbox"/>	TerminateInstances	December 03, 2022, 23:11:38 (U...)	Aakash	ec2.amazonaws.com	AWS::EC2::Instance	i-
<input type="checkbox"/>	StopInstances	December 03, 2022, 23:04:48 (U...)	root	ec2.amazonaws.com	AWS::EC2::Instance	i-
<input type="checkbox"/>	SendSSHPublicKey	December 03, 2022, 22:41:29 (U...)	root	ec2-instance-connect.amazonaws.com	AWS::EC2::Instance	i-
<input type="checkbox"/>	StartInstances	December 03, 2022, 22:40:25 (U...)	root	ec2.amazonaws.com	AWS::EC2::Instance	i-
<input type="checkbox"/>	AssociateElbInstance...	December 03, 2022, 20:45:03 (U...)	Aakash	ec2.amazonaws.com	AWS::EC2::Instance	i-
<input type="checkbox"/>	AssociateElbInstance...	December 03, 2022, 20:44:53 (U...)	Aakash	ec2.amazonaws.com	AWS::EC2::Instance	i-
<input type="checkbox"/>	AssociateElbInstance...	December 03, 2022, 20:44:46 (U...)	Aakash	ec2.amazonaws.com	AWS::EC2::Instance	i-
<input type="checkbox"/>	AssociateElbInstance...	December 03, 2022, 20:44:38 (U...)	Aakash	ec2.amazonaws.com	AWS::EC2::Instance	i-

Hence, Cobra Kai can maintain a constant flow of operations and logs efficiently and the Security team can track in case any attack is waged against their resources too.

### ***Initial Setup of the AWS RDS Replication for High Availability (Relational Database Service)***

AWS RDS is a Distributed Relational Database Service that Amazon uses for setting up a Relational Database in the Cloud.

As Cobra Kai maintains a Database of all its customers along with their credit card data, I have created a Primary RDS Database and replicated that to another Availability Zone for high availability for Disaster Recovery and Business Continuity.

The environment I have created is a **Multi-AZ (Availability Zone)** environment where I have added Bert, the System Administrator of Cobra Kai to have access to the Database. Bert would

need to have constant interaction with the Database to retrieve valuable customer information and perform routine operations. He is a highly technical person too.

I have also connected the Database to our initial Subnets for easy Replication as shown below:

The screenshot shows the AWS RDS console for a MySQL database instance named "cobrakai-rds-primary".

**Summary** tab details:

DB identifier cobrakai-rds-primary	CPU 2.27%	Status Available	Class db.t3.micro
Role Instance	Current activity 0 Connections	Engine MySQL Community	Region & AZ us-east-1a

**Connectivity & security** tab details:

Endpoint & port	Networking	Security
Endpoint cobrakai-rds-primary.cnwv4zfouerh.us-east-1.rds.amazonaws.com	Availability Zone us-east-1a	VPC security groups <a href="#">Web Server 1-CobraKai SG (sg-0cc3929e8b1cccd686)</a>
Port 3306	VPC <a href="#">Cobra Kai-VPC-Aakash (vpc-062862b2fab0482fe)</a>	Active
	Subnet group default-vpc-062862b2fab0482fe	Publicly accessible No
	Subnets <a href="#">subnet-0b223953506659338</a> <a href="#">subnet-0218816640a0e9bbd</a> <a href="#">subnet-00434eea4ba34da30</a> <a href="#">subnet-0f41b184b46566676</a>	Certificate authority rds-ca-2019
	Network type IPv4	Certificate authority date August 22, 2024, 13:08 (UTC-04:00)

The below screenshot shows the Primary RDS Server that will be replicated in another Availability Zone. I have also encrypted the Primary RDS Server with the **AWS/RDS KMS Key** as shown below.

**cobrakai-rds-primary**

**Summary**

DB identifier cobrakai-rds-primary	CPU <div style="width: 2.27%;">2.27%</div>	Status Available	Class db.t3.micro
Role Instance	Current activity <div style="width: 0%;">0 Connections</div>	Engine MySQL Community	Region & AZ us-east-1a

Connectivity & security | Monitoring | Logs & events | **Configuration** | Maintenance & backups | Tags

**Instance**

Configuration	Instance class	Storage	Performance Insights
DB instance ID cobrakai-rds-primary	Instance class db.t3.micro	Encryption Enabled	Performance Insights enabled Turned off
Engine version 8.0.28	vCPU 2	AWS KMS key aws/rds	
DB name CobraKaiRDSPrimary	RAM 1 GB	Storage type General Purpose SSD (gp2)	
License model General Public License	Availability	Storage 200 GB	
Option groups default:mysql-8-0 <span style="color: green;">Sync</span>	Master username Bert	Provisioned IOPS -	
Amazon Resource Name (ARN) arn:aws:rds:us-east-1:58489529881:db:cobrakai-rds-primary	IAM DB authentication Not enabled	Storage throughput -	
Resource ID db:cobrakai-rds-primary	Multi-AZ Yes	Storage autoscaling Enabled	

## Recommendations for Backing Cobra Kai's Resources using AWS Backups

Backing up resources is a strong Security practice and is always recommended when companies face a cyber or natural-related attack. In the Cloud, AWS uses AWS Backups to perform this.

AWS Backups are a cost-effective way of backing up your Cloud resources using a fully-managed, policy-based service that simplifies Data Protection and is a good practice to always maintain high availability.

1. AWS Backups allow the creation of periodic backups of the resources of our choice. I tried this approach on Web Server 1 & App Server 1 of Cobra Kai. I also backed up the EBS Volume, “**vol-0bdd0fe99aaf6ddcf**” as shown by the below screenshot.

**Jobs**

In jobs, you can monitor the status and other details of backup, restore, and copy activity.

**Backup jobs** | Restore jobs | Copy jobs

**Backup jobs info**

Records of your scheduled or on-demand backups.

Q Filter backup jobs by job ID, status, resource ID or resource type

Backup job ID	Status	Resource ID	Resource type	Creation time	Start by
0e6a23a0-ae53-4008-a3c8-f137142b730a	Completed	volume/vol-0bdd0fe99aaf6ddcf	EBS	December 6, 2022, 13:51:03 (UTC-05:00)	December 6, 2022, 14:51:03 (UTC-05:00)
26533093-7ca2-42d5-b79c-2a3f9951e5d7	Completed	instance/i-09e4b6e6f58117e6f	EC2	December 6, 2022, 13:41:22 (UTC-05:00)	December 6, 2022, 14:41:22 (UTC-05:00)

2. The below screenshots show the Backup Plan I have created for backing up the instance's Daily and Monthly Backup rules.

Backup plan name	Last runtime	Last modified
CobraKai_Backup_Plan_Aakash	-	December 6, 2022, 13:52:13 (UTC-05:00)

Backup plan name	Version ID	Last modified	Last runtime
CobraKai_Backup_Plan_Aakash	NmEzNGZhOWUtzZWU4MC00Y2MyLWl4NmItYzNIYTcwOTMwODE1	December 6, 2022, 13:58:33 (UTC-05:00)	-

Name	Backup vault	Destination Backup vault
DailyBackups	CobraKaiBackupVault	-
Monthly	CobraKaiBackupVault	-

Name	IAM role ARN	Creation time
CobraKaiBackup_resources	arn:aws:iam::584895298881:role/service-role/AWSBackupDefaultServiceRole	December 6, 2022, 13:49:19 (UTC-05:00)

3. To protect the backups that are taken, AWS uses the concept of “**Backup Vaults**”. Backup Vaults are encrypted using a **KMS Encryption Key** which is depicted by the screenshot below.

Backup vault name	Vault lock status	Recovery points	KMS encryption key ID
CobraKaiBackupVault	-	1	b5c4dfcd-3466-4eed-871a-dad51ef36cfe
Default	-	1	b5c4dfcd-3466-4eed-871a-dad51ef36cfe

Hence, Cobra Kai can use AWS Backups to back up their resources and encrypt those backups too all in the Cloud.

## ***Recommendations for Handling Network Firewalls using Security Groups & NACLs (Network Access Control Lists)***

Security Groups help to control the traffic that is allowed to enter or leave that instance. Security Groups are a good way to control traffic as they are the equivalent of a Network Firewall, but work specifically on the Cloud. When you create a VPC, it comes with a default Security Group. They are **implicit**, **stateful**, and **generic** hence it's an easy way to enforce basic Network Security on the Cloud.

I have created the Security Group “**Web Server 1-CobraKai SG**” for all the EC2 Instances.

1. The below screenshot depicts the Inbound Rules for the Security Group. I have added SSH & HTTP connections from only “**My IP**” so that connections are securely made to the Server and are not directly accessible by the Internet. The HTTPS connection is open to the public as it is HTTP over a Secure channel.

The screenshot shows the AWS EC2 Security Groups console. At the top, there are tabs for 'Inbound rules', 'Outbound rules', and 'Tags'. Below the tabs, a message says 'You can now check network connectivity with Reachability Analyzer' with a 'Run Reachability Analyzer' button. The main area displays the 'Inbound rules (3)' table:

Security group rule...	IP version	Type	Protocol	Port range	Source	Description
sgr-00eac2bd11b9be7...	IPv4	SSH	TCP	22	129.2.180.44/32	SSH Connection for W...
sgr-0bbdddfb7221dc8de	IPv4	HTTP	TCP	80	129.2.180.44/32	HTTP Connection for ...
sgr-0eed000c7cd50e4bd	IPv4	HTTPS	TCP	443	0.0.0.0/0	HTTPS Connection for ...

2. The below screenshot shows the Outbound Rules for the Security Group. In this case, I have kept the default Security Group rules where all connections can be broadcasted.

The screenshot shows the AWS EC2 Security Groups page. The security group selected is "sg-0cc3929e8b1ccd686 - Web Server 1-CobraKai SG". The "Outbound rules" tab is active, displaying one rule:

Name	Security group rule...	IP version	Type	Protocol	Port range	Destination	Description
-	sgr-033995d00225f52...	IPv4	All traffic	All	All	0.0.0.0/0	-

Network ACLs are an optional layer of Security in the Network that controls traffic too. They are **explicit, stateless**, and more **specific** than Security Groups. They are also a form of Firewall and follow the “**Implicit Deny Principle**”, and the order of Precedence is from **Top to Bottom**.

I have configured a NACL and configured Inbound Rules that allow custom TCP traffic over a particular port range from 1024 - 65535.

The screenshot shows the AWS VPC Network ACLs page. The network ACL selected is "acl-079713d8715259850". The "Inbound rules" tab is active, displaying two rules:

Rule number	Type	Protocol	Port range	Source	Allow/Deny
100	Custom TCP	TCP (6)	1024 - 65535	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

I have configured Outbound Rules that allow all traffic to leave the network as shown below.

The screenshot shows the AWS VPC Network ACL configuration page for a specific Network ACL (acl-079713d8715259850). The 'Outbound rules' tab is selected. There are two rules listed:

Rule number	Type	Protocol	Port range	Destination	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	<span style="color: green;">Allow</span>
*	All traffic	All	All	0.0.0.0/0	<span style="color: red;">Deny</span>

A message at the top indicates: "You can now check network connectivity with Reachability Analyzer".

The below screenshot shows the Subnet Associations for the NACL.

The screenshot shows the AWS VPC Network ACL configuration page for the same Network ACL (acl-079713d8715259850). The 'Subnet associations' tab is selected. There are four subnet associations listed:

Name	Subnet ID	Associated with	Availability Zone	IPv4 CIDR	IPv6 CIDR
Public Subnet 2-CobraKai	subnet-0218816640a0e9bbd	acl-079713d8715259850	us-east-1b	129.2.1.0/24	-
Public Subnet 1-CobraKai	subnet-0b223953506659338	acl-079713d8715259850	us-east-1a	129.2.0.0/24	-
Private Subnet 2-RDS Replication	subnet-0f41b184b46566676	acl-079713d8715259850	us-east-1b	129.2.3.0/24	-
Private Subnet 1-Amazon RDS	subnet-00434eeaa4ba34da30	acl-079713d8715259850	us-east-1a	129.2.2.0/24	-

Hence, by incorporating Security Groups & NACLs, Cobra Kai can control Network Traffic entering and leaving the Network and also use secure protocols such as SSH to establish Remote Connections. Other protocols, like SFTP, can also be incorporated in the future in the same manner.

## **Recommendations for Handling Identity & Access Management**

Identity & Access Management is a critical domain for providing security in the Cloud. AWS has an IAM Dashboard that allows creating “**User Groups**”, “**Roles**” & attach “**Policies**” to those users. This way the “**Principle of Least Privilege**” will be followed, which is a strong practice both in traditional and Cloud environments.

I have created User Groups for all the employees of Cobra Kai as shown below screenshot:  
The User Groups include:

- **Admin**
- **Developer**
- **Management**
- **Security\_and\_Risk\_Management**

These are shown in the below screenshot:

The screenshot shows the AWS IAM 'Users' page. At the top, there's a banner with the message "Ready to streamline human access to AWS and cloud apps?". Below the banner, it says "Identity Center is enabled. We recommend managing workforce users' access to AWS accounts and cloud applications in Identity Center." There are links to "Learn more" and "Watch how it works". The main table lists 7 users:

User name	Groups	MFA	Password age	Active key age	ARN	Create
Aisha_Robinson	Security_and_Risk_Management	Virtual	9 days ago	9 days ago	arn:aws:iam::584895298881:user/Aisha_Robinson	9 day:
Bert	Admin	Virtual	9 days ago	9 days ago	arn:aws:iam::584895298881:user/Bert	9 day:
Demetri	Developer	Virtual	9 days ago	9 days ago	arn:aws:iam::584895298881:user/Demetri	9 day:
ElliHawk_Moskowitz	Developer	Virtual	9 days ago	9 days ago	arn:aws:iam::584895298881:user/ElliHawk_Moskowitz	9 day:
Johnny_Lawrence	Management	Virtual	9 days ago	9 days ago	arn:aws:iam::584895298881:user/Johnny_Lawrence	9 day:
Miguel_Diaz	Management	Virtual	9 days ago	9 days ago	arn:aws:iam::584895298881:user/Miguel_Diaz	9 day:

1. I have added **Bert**, the **System Administrator**, to the **Admin** group because he would need to interact with the Databases & Servers regularly along with his technical knowledge. No other employee would require that much access to perform his/her daily functions. He has the policies of “**AdministratorAccess**”, “**AmazonRDSFullAccess**”, “**DatabaseAdministrator**” and “**SystemAdministrator**”. This is shown in the below screenshot.

The screenshot shows the AWS IAM User Groups interface. The Admin user group is selected. The 'Permissions' tab is active. Under 'Permissions policies (4)', there are four AWS managed policies listed:

Policy name	Type	Description
AmazonRDSFullAccess	AWS managed	Provides full access to Amazon RDS via the AWS Management Console.
DatabaseAdministrator	AWS managed - job functi...	Grants full access permissions to AWS services and actions required to set up and configure AWS database serv...
SystemAdministrator	AWS managed - job functi...	Grants full access permissions necessary for resources required for application and development operations.
AdministratorAccess	AWS managed - job functi...	Provides full access to AWS services and resources.

2. I have added **Eli Hawk Moscowitz**, the **Chief Information Officer & Demetri**, the **Web Developer** in the **Developer** group, and given them the **“AWSCodeBuildDeveloperAccess”** because both of them require access to a Development environment to perform their daily activities.

The screenshot shows the AWS IAM User Groups interface. The Developer user group is selected. The 'Permissions' tab is active. Under 'Permissions policies (1)', there is one policy listed:

Policy name	Type	Description
AWSSCodeBuildDeveloperAccess	AWS managed	Provides access to AWS CodeBuild via the AWS Management Console, but does not allow CodeBuild project administration. Also attach AmazonS3ReadOnlyAccess to provide access to download build artifacts.

The policy details show the JSON structure:

```

1- {
2-   "Statement": [
3-     {
4-       "Action": [
5-         "codebuild:StartBuild",
6-         "codebuild:StopBuild",
7-         "codebuild:StartBuildBatch",
8-         "codebuild:StopBuildBatch",
9-         "codebuild:RetryBuild",
10-        "codebuild:RetryBuildBatch",
11-        "codebuild:BatchGet",
12-        "codebuild:GetResourcePolicy",
13-        "codebuild:DescribeTestCases",
14-        "codebuild:DescribeCodeCovverages",
15-        "codebuild>List",
16-        "codecommit:GetBranch",
17-        "codecommit:GetCommit",
18-        "codecommit:ListCommits"
19-      ],
20-      "Effect": "Allow",
21-      "Resource": "*"
22-    }
23-  ]
24-}
  
```

3. I have added **Johnny Lawrence**, the Chief Executive Officer, and **Miguel Diaz**, the Chief Operating Officer in the **Management** group and they have the policy of "**CobraKaiAccountManagement**". As these are C-Level executives they are kept in a common group and also they are Non-Technical but must be in charge of knowing who has how much access to data.

The screenshot shows the AWS IAM User Groups Management interface. The 'Management' group is selected. The 'Permissions' tab is active, showing one managed policy named 'CobraKaiAccountManagement'. The policy document is as follows:

```

1 - {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": [
8         "account:GetContactInformation",
9         "account:GetAlternateContact"
10      ],
11      "Resource": "arn:aws:account::584895298881:account"
12    }
13  ]
14 }

```

4. I have added **Aisha Robinson**, the Chief Information Security Officer (CISO) in the **Security\_and\_Risk\_Management** group and she has 3 policies, namely, "**CloudWatchLogsFullAccess**", "**CloudWatchReadOnlyAccess**" and "**AWSCloudTrailFullAccess**". This is because the CISO should have full access to the logs and dashboards of CloudWatch and CloudTrail but need not perform any setup or implementation with those logs. This is shown in the below screenshot.

The screenshot shows the AWS IAM User Groups interface. At the top, there's a navigation bar with 'IAM' and 'User groups'. Below it, the group name 'Security\_and\_Risk\_Management' is displayed. On the right, there are 'Delete' and 'Edit' buttons. A 'Summary' section shows details: 'User group name: Security\_and\_Risk\_Management', 'Creation time: November 27, 2022, 18:38 (UTC-05:00)', and 'ARN: arn:aws:iam::584895298881:group/Security\_and\_Risk\_Management'. Below this, tabs for 'Users', 'Permissions' (which is selected), and 'Access Advisor' are visible. The 'Permissions' tab shows 'Permissions policies (3) Info' with a note about attaching up to 10 managed policies. A search bar says 'Filter policies by property or policy name and press enter.' Below is a table with columns 'Policy name', 'Type', and 'Description'. The table contains three rows:

Policy name	Type	Description
CloudWatchLogsFullAccess	AWS managed	Provides full access to CloudWatch Logs
CloudWatchReadOnlyAccess	AWS managed	Provides read only access to CloudWatch.
AWSCloudTrail_FullAccess	AWS managed	Provides full access to AWS CloudTrail.

By following these practices, Cobra Kai can implement strong IAM Policies within their Cloud environments to prevent both insider threats and maintain proper audits of all their user's actions.

## ***Conclusion***

Overall, the process of designing all the required recommendations and the architecture in the AWS Cloud Environment can be a bit challenging for Cobra Kai, especially during its initial stages, but almost all of its problems can be easily solved in a very small time frame. Hence, by following a proper Security Baseline, Best Practices, and Compliance Frameworks, Cobra Kai can truly leverage the Cloud to its full capacity and boost its business!

## **References**

These are the references I have used for this Project in APA Format:

- AWS, A. (n.d.). *AWS Management and governance tools workshop*. AWS Management and Governance Tools Workshop. Retrieved December 8, 2022, from [https://mng.workshop.aws/ssm/use-case-labs/inventory\\_patch\\_management/patch.html](https://mng.workshop.aws/ssm/use-case-labs/inventory_patch_management/patch.html)
- Team, W.-A. (n.d.). *Cloudfront with S3 Bucket Origin*. CloudFront with S3 Bucket Origin :: AWS Well-Architected Labs. Retrieved December 8, 2022, from [https://www.wellarchitectedlabs.com/security/100\\_labs/100\\_cloudfront\\_with\\_s3\\_bucket\\_origin/](https://www.wellarchitectedlabs.com/security/100_labs/100_cloudfront_with_s3_bucket_origin/)
- Ribeiro, A. (2021, July 4). *Build a serverless website using Amazon S3 and Route 53*. Medium. Retrieved December 8, 2022, from <https://towardsdatascience.com/build-a-serverless-website-using-amazon-s3-and-route-53-c741fae6ef8d>
- Team, W.-A. (n.d.). *Create CloudWatch dashboard*. Create CloudWatch Dashboard :: AWS Well-Architected Labs. Retrieved December 8, 2022, from [https://www.wellarchitectedlabs.com/performance-efficiency/100\\_labs/100\\_monitoring\\_li\\_nux\\_ec2\\_cloudwatch/3\\_creating\\_cloudwatch\\_dashboard/](https://www.wellarchitectedlabs.com/performance-efficiency/100_labs/100_monitoring_li_nux_ec2_cloudwatch/3_creating_cloudwatch_dashboard/)
- KnowledgeHut. (2020, January 2). *AWS relational database- how to create RDS instance in 11 easy steps*. Knowledgehut. Retrieved December 8, 2022, from <https://www.knowledgehut.com/tutorials/aws/creation-of-rds-instance>
- Conn, M., & Conn, M. (2021). *Getting started*. Amazon. Retrieved December 8, 2022, from <https://aws.amazon.com/getting-started/hands-on/amazon-ec2-backup-and-restore-using-aws-backup/>

