WeCasa

Network Design Document

Team HAGS JP

Team Lead:

Allison Austin

Team Members:

Githel Lynn Suico

Haley Nguyen

Joshua Quibin

Judy Li

Matthew Chung

Date Submitted: November 9, 2022

Github Repository:

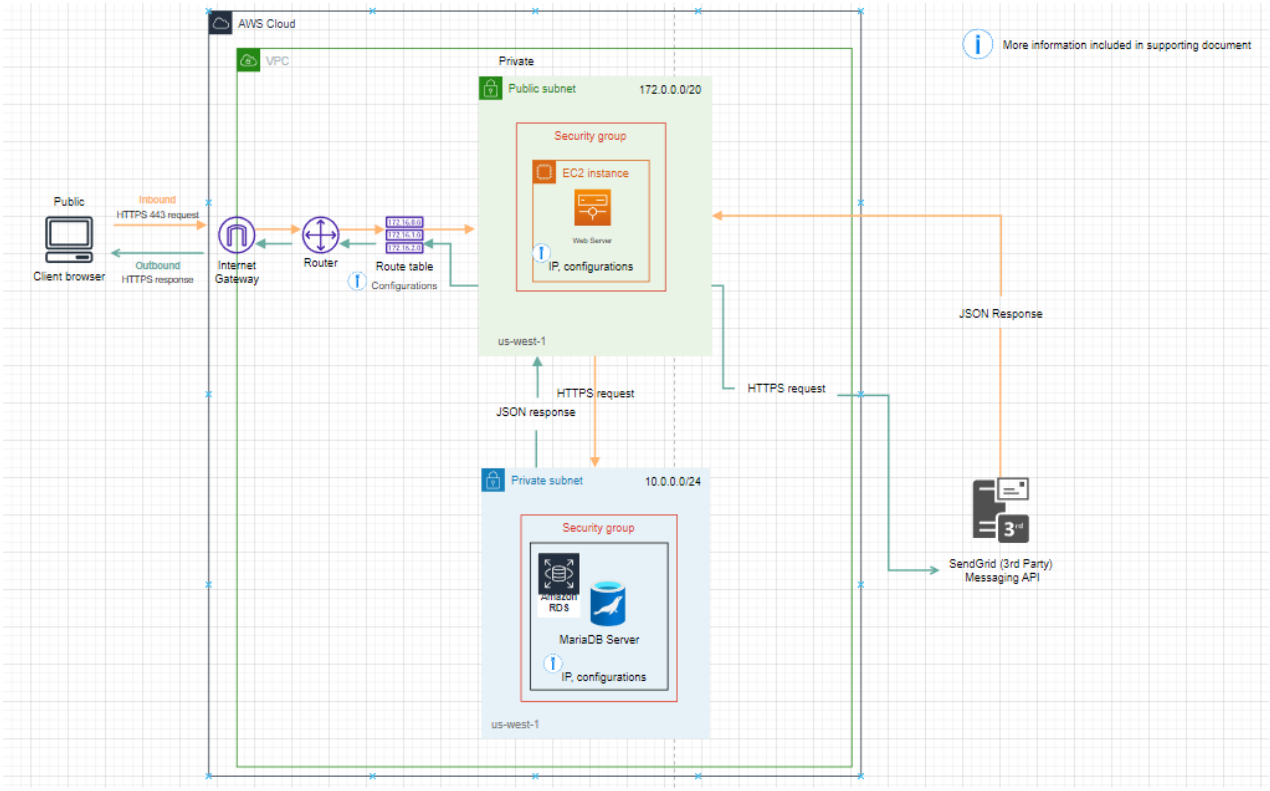
https://github.com/githelsui/WeCasa

Network Design Version Table

Version	Description	Date
1.0	Initial Network Design - Diagram - Components - Flow of Traffic	11/9/2022

Table of Contents

Diagram



Hardware Components

Network Appliances

WeCasa will be utilizing hardware provided by Amazon Web Services to host the application. AWS is responsible for WeCasa's application/database servers, routers, gateway, and other network appliances. Our DMZ/firewall layer will be handled by our cloud provider (AWS), and since we will be controlling traffic to and from our VPC with a custom route table and security groups for our EC2 instances, we will not need a load balancer. Since WeCasa is in its earliest stages and has not released its first version of the application, we will not be using a Web Farm to host multiple versions for development and testing. All WeCasa users will be interacting with the same version hosted on a single web server.

Software Components

Client/Browser

To use WeCasa, the user must access the application on a Google Chrome Browser version 104 and above. The client can then make HTTPS requests on port 443, which is the standard port for HTTPS traffic that connects the Chrome browser to the WeCasa web server. Any data that travels on port 443 is encrypted using SSL certificates.

Virtual Private Cloud

Amazon Virtual Private Cloud (VPC) will allow us to create a default private network hosted within AWS giving us control over the connectivity and security of our resources. The VPC increases security by isolating the encapsulated backend components on AWS Cloud. To control web traffic to our VPC endpoint and limit outbound HTTPS traffic, we will configure the VPC Route table and utilize security groups for our EC2 instances. We will use the default VPC that is associated with our AWS account. Default VPCs include an Internet Gateway, a Route Table, and a public IPv4 address that will be assigned to our EC2 instance in our public subnet (docs.aws.amazon.com). The range of IP addresses for our public subnet is 172.0.0.0/20 and the IP address that will be assigned to our EC2 instance inside our public subnet is 172.0.0.1. The range of IP addresses for our private subnet is 10.0.0.0/24 and the IP address that will be assigned to our RDS instance inside this subnet is 10.0.0.1.

Our custom Route Table for our VPC will have the following configurations:

Destination	**Target**
172.0.0.0/20	local
0.0.0.0/0	igw-id

Internet Gateway

Our default VPC will include an **Internet Gateway** that will allow our subnets to connect to the internet and receive IPv4 traffic (Amazon.com). The IGW will only be connected to our public subnet within our VPC, because connecting our private subnet to the Internet would require a NAT Gateway, which charges hourly per use. Since we are exclusively using resources in the free tier of AWS, we will **not** be using the NAT Gateway

service, and our private subnet will not be accessible from the Internet (only through port SSH 22 and our public EC2 instance). Setting up our IGW would require our public subnet to have a public IPv4 address, and this will be configured when we launch our EC2 instance within our VPC.

Elastic Compute Cloud

Within our VPC, we will have **Amazon EC2 Instances** to launch our servers virtually. Amazon EC2 allows us to offload the physical hardware management of our servers while still being able to configure security and monitor traffic/storage (docs.aws.amazon.com). Because WeCasa will remain under the free tier for AWS, we will be using one t2.micro EC2 instance for our application. This instance will be public and contain our web server. This instance will be accessible from the Internet via the IGW and can communicate with third-party APIs, other instances, and the client. This instance will have its own Security Group with custom rules for accepted protocols and port ranges.

Our EC2 server will have the following configurations:

Virtual CPU	**RAM**	**SSD**
1	1 GB	30 GB

Security Groups

By default, Amazon EC2 instances use **AWS Security Groups** as a firewall to specify ports, IP ranges, and ports. Designing an AWS Security Group for maximum security will minimize risk by locking down network ports for intra-node communication and limiting external access to network ports for application access and administrative control (Amazon.com). Both our EC2 instance and our RDS instance will have configured Security Groups.

Instance	**Protocols**	**Ports**	**Rules**
EC2	TCP/IP, TLS	443, 22	This instance will allow inbound/outbound HTTPS access from all IPv4 addresses and SSH port 22.
RDS	TCP/IP, TLS	22	This instance will allow inbound HTTPS access from the web server and accept requests made on SSH port 22. This

			instance will allow outbound traffic to all local destination IP addresses in the VPC route table and any responses sent on SSH port 22.

Relational Database Service

Amazon Relational Database Service (RDS) will allow us to set up, operate, and scale our databases in the cloud (*Amazon Web Services*, 2022). This RDS instance will reside in our private subnet and will not be connected to an EC2 compute resource.

Our RDS instance will have the following configurations:

Engine Type/Version	**Virtual CPU**	**Allocated Storage**	**Authentication**	**Database Port**
MariaDB 10.6.11	1	RAM: 1GB SSD: 50GB	IAM	3307

Third-Party APIs

WeCasa uses a third-party API to send notifications via SMS and email. This API is provided by Twillio's SendGrid, and we will be using the SendGrid Web API version 3. Messages can be sent to this API via a curl command (POST or PUT) with a JSON payload. This command will send an HTTPS request to the API, which will respond in JSON format. The response code and content-type header will allow us to interpret the response and handle any errors in the Business Logic Layer.

Flow of Traffic

Inbound traffic will come in as HTTPS requests from the client on a browser. These requests will be made on port 443 and will be routed to the VPC by the Internet Gateway. Since there is only one VPC being used, the IGW will either accept the request and move it to the application's Entry Point Layer, or deny the request and return a 400 response status code.

After being accepted through the Entry Point Layer, the requests get sent to our public Amazon EC2 Instance which contains our web server. Within this instance, security is configured and traffic/storage is monitored. To ensure maximum security, the EC2

instance will only allow inbound HTTPS access from IPv4 addresses and SSH port 22. Furthermore, our private RDS will accept inbound HTTPS access from the web server within our EC2 instance, and accept requests made on SSH port 22. If the request is valid and successful, our RDS will return a JSON response back to our public web server. To ensure security in this layer, similarly to our web server, Security Groups will be configured to deny any unauthorized requests. After the JSON response gets sent over to the web server, an outbound HTTPS response will be received by our client. After being received by the client, this JSON response will be parsed properly and the returned data will be presented to the user through the interface or an updated view.

From the web server, HTTPS requests in the form of a curl command will be sent to a third-party API, SendGrid, in order to send notifications via SMS and email. Once the request is received, the SendGrid API will send a JSON response back to the web server with either successful data or an error.

Glossary

Term	**Definition**
API	Application Programming Interface
AWS	Amazon Web Services
Client	Any user that is making requests to the WeCasa web server through the presentation layer.
EC2	Elastic Compute Cloud
IGW	Internet Gateway
NAT	Network Address Translation
RDS	Relational Database Service
SSL	Secure Sockets Layer
TLS	Transport Layer Security
VPC	Virtual Private Cloud

References

AWS VPCs. (2022). Docs.aws.amazon.com. https://docs.aws.amazon.com/vpc/latest/userguide/how-it-works.html

Connect to the internet using an internet gateway - Amazon Virtual Private Cloud. (2022). Amazon.com. https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html

Control traffic to resources using security groups - Amazon Virtual Private Cloud. (2022). Amazon.com. https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html

EC2 Infrastructure Security. (2022). Docs.aws.amazon.com. https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/infrastructure-security.html

EC2 setup. (2022). Docs.aws.amazon.com. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

Fully Managed Relational Database - Amazon RDS - Amazon Web Services. (2022). Amazon Web Services, Inc. https://aws.amazon.com/rds/