Contents

[Question 1 (Creating an origin server) 3](#_Toc155912496)

[Step-by-Step Implementation/Methodologies 3](#_Toc155912497)

[Creating an account in AWS to serve as an origin 3](#_Toc155912498)

[Preparation of webserver and its environment 5](#_Toc155912499)

[How did you fill the gaps (if any) in your knowledge during the process? 6](#_Toc155912500)

[Question 2 (Enabling proxy traffic to Cloudflare) 6](#_Toc155912501)

[What use cases can you see different products being useful for? 8](#_Toc155912502)

[How do you imagine that a target customer will find this experience? 8](#_Toc155912503)

[Question 3 (Securing Communication) 9](#_Toc155912504)

[Method 1 (Encrypting origin with Let’s Encrypt) 9](#_Toc155912505)

[Method 2 (Encrypting origin with ZeroSSL) 9](#_Toc155912506)

[Enabling Full-Strict mode on Cloudflare 11](#_Toc155912507)

[What use cases can you see different products being useful for? 11](#_Toc155912508)

[How did you fill the gaps (if any) in your knowledge during the process? 12](#_Toc155912509)

[How do you imagine that a target customer will find this experience? 12](#_Toc155912510)

[Question 4 (Rate Limiting) 13](#_Toc155912511)

[Examples of configurations with custom rules 13](#_Toc155912512)

[Example of configuration with rate limitations 14](#_Toc155912513)

[What use cases can you see different products being useful for? 15](#_Toc155912514)

[How did you fill the gaps (if any) in your knowledge during the process? 16](#_Toc155912515)

[How do you imagine that a target customer will find this experience? 16](#_Toc155912516)

[Question 5 (Cloudflare Tunnel) 17](#_Toc155912517)

[Implementation 17](#_Toc155912518)

[What use cases can you see different products being useful for? 19](#_Toc155912519)

[How did you fill the gaps (if any) in your knowledge during the process? 19](#_Toc155912520)

[How do you imagine that a target customer will find this experience? 19](#_Toc155912521)

[Question 6 (SSO Idp) 20](#_Toc155912522)

[Implementation 20](#_Toc155912523)

[What use cases can you see different products being useful for? 26](#_Toc155912524)

[How did you fill the gaps (if any) in your knowledge during the process? 26](#_Toc155912525)

[How do you imagine that a target customer will find this experience? 26](#_Toc155912526)

[Question 7 (Lock Down Tunnel Access) 27](#_Toc155912527)

[Implementation 27](#_Toc155912528)

[What use cases can you see different products being useful for? 31](#_Toc155912529)

[How did you fill the gaps (if any) in your knowledge during the process? 32](#_Toc155912530)

[How do you imagine that a target customer will find this experience? 32](#_Toc155912531)

[Question 8 (Cloudflare Worker) 33](#_Toc155912532)

[Implementation 33](#_Toc155912533)

[Sub-Question 1 33](#_Toc155912534)

[Sub-Question 2 35](#_Toc155912535)

[Sub-Question 3 36](#_Toc155912536)

[Method 1 (That failed) 36](#_Toc155912537)

[Method 2 (Which worked but compromises the requirement) 37](#_Toc155912538)

[What use cases can you see different products being useful for? 39](#_Toc155912539)

[How did you fill the gaps (if any) in your knowledge during the process? 39](#_Toc155912540)

[How do you imagine that a target customer will find this experience? 40](#_Toc155912541)

# Question 1 (Creating an origin server)

Create an origin web server on a platform of your choosing. This could be in AWS, Google Cloud, DigitalOcean, your Raspberry Pi, etc.

* 1. This web server must run an endpoint that returns all HTTP request headers in the body of the HTTP response.
  2. The web server can be something that you have written yourself (e.g. in JavaScript, Python, etc) or by using a 3rd party application.

## Step-by-Step Implementation/Methodologies

### Creating an account in AWS to serve as an origin

1. Signed up account in AWS (<https://portal.aws.amazon.com/billing/signup>)
2. Created a new instance under EC2 product Amazon Elastic Compute Cloud (Amazon EC2) is a web service provided by Amazon Web Services (AWS) that offers resizable compute capacity in the cloud, there is a free tier that although with low hardware limitations, as well as limited functionality, the free tier will be sufficient to run the website.
3. Create a new instance with “Launch instances” and choose the appropriate selection.  
   A screenshot of a computer

   Description automatically generated
   1. Name : *Insert Name of the Web Server*
   2. Application and OS Images : *Choose the OS image preferred for the instance. However I choose* ***Ubuntu LTS*** *since it was the most user friendliest and most familiar to me*A screenshot of a computer

      Description automatically generated
   3. Network Security : *I allowed HTTP and HTTPS in addition to no limitation of SSH traffic*A screenshot of a computer

      Description automatically generated
   4. Configure Storage : *This will be the storage configuration for the instance, free users are eligible for 30GB of storage, however I opt for a 20 GB configuration with no partition*A screenshot of a computer screen

      Description automatically generated
4. Proceed to create the new instance with “Launch instance”.

### Preparation of webserver and its environment

1. The purchase of Domain
   1. The purchase of the domain is from GoDaddy (<https://www.godaddy.com/en-ph>), where the domain name “seanproject2024.xyz” was purchased as the domain is available.
2. Signing up for a free tier cloudflare account to enable acceleration and security products
3. Setting up ufw to enable ports in Ubuntu origin
   1. sudo ufw allow 80/tcp <🡨 This enables TCP port through HTTP common port 80
   2. sudo ufw allow 443/tcp 🡨 This enables TCP port through HTTPS common port 443
   3. sudo ufw allow 22/ssh 🡨 This enables SSH session, if not included, the instance cannot be connected
   4. sudo ufw enable 🡨 The command enables linux’s simple firewall configuration
4. Downloading and setting up NGINX
   1. Update the system with the command   
      sudo apt update && sudo apt upgrade -y && sudo apt autoremove -y && sudo apt clean
   2. Install required Packages   
      sudo apt install ntp sed git curl zip unzip nginx redis certbot ufw php-fpm php-curl php-mysql php-mbstring php-xml php-gd php-redis php-zip php-imagick php-bcmath php-intl php-tokenizer -y
   3. Running the NGINX service  
      sudo systemctl start nginx 🡨 Starting the daemon  
      sudo systemctl enable nginx 🡨 Starting the daemon  
      sudo systemctl status nginx 🡨 Checking the status of the daemon  
      sudo systemctl reload nginx 🡨 This command is required to reload nginx to apply the latest change
   4. Configuration file paths
      1. The main NGINX configuration file is /etc/nginx/nginx.conf, and site-specific configurations are typically stored in the /etc/nginx/sites-available/ directory
      2. The default configuration files for NGINX are typically located in /etc/nginx/
      3. The default web root is usually set to /var/www/html/ , therefore index.html will be placed under this directory  
         \*Set the correct permission and user assign for the html file with chmod to ensure that NGINX can read directory file   
         sudo chown -R www-data:www-data \*

Sudo chmod 755 html

* 1. Testing NGINX configurations to ensure that there is no conflicts  
     sudo nginx -t

1. Configuring NGINX to point to the domain as well link HTTPS to the file in which the cert will be stored.
2. Creating a simple HTML file as a website for the domain to demonstrate content.

### How did you fill the gaps (if any) in your knowledge during the process?

I am used to XAMPP environment, and therefore NGINX which could be considered an industry-wide standard is something that is foreign to me, most of my time is spent researching on how to configure and enable nginx services In my web origin. HTML and CSS is not something new to me since I have a degree of experience since my college years. The gap in knowledge is satiated with knowledge from blogs and website guides that assisted and taught me how NGINX should work.

# Question 2 (Enabling proxy traffic to Cloudflare)

Proxy traffic to this server through Cloudflare. Add necessary configurations on Cloudflare.

1. After the creation of a free tier cloudflare account, I have added my domain “seanproject2024.xyz” under the **Websites** section of Cloudflare Dashboard (https://dash.cloudflare.com/c61f14cea4468755ca216500f2870e99)  
   A screenshot of a computer

   Description automatically generated
2. Have your domain directed to the Cloudflare’s nameserver in the respective domain provider that you have purchased from, have the domain point to the specified cloudflare nameservers provided in the dash, it can be found under **DNS > Records**  
   A screenshot of a computer

   Description automatically generated  
   This will enable authoritative DNS to point your domain to Cloudflare’s DNS/Service IPs  
     
   In my case, I have configured in GoDaddy to point to Cloudflare’s Nameserver under **Domain > DNS > Nameserver**  
   A screenshot of a computer

   Description automatically generated
3. Configure to have your domain point to the origin of your server  
     
   A screenshot of a computer

   Description automatically generated  
     
   The origin of the IP is the **Public IPv4 IP** that can be found in the Amazon EC2 instance

### What use cases can you see different products being useful for?

1. the DNS nameserver is queried to find the corresponding IP address of the web server hosting that website, it also masks the Real IP of the origin web server
2. DNS nameservers can be used for domain parking to display placeholders or cached content
3. DNS in CDNs can optimize user experience by reducing latency of content for user requests from different geolocation. Optimizing resource utilization and high availability while reducing possible network limitations when requesting content from a server physically far from the user
4. DNS enable load balancing for Cloudflare when handling ingress and egress content traffic
5. Enabling subdomain management that points to different origin IPs
6. Enable DNS Security (DNSSEC) for security measures as it adds an additional layer of security by verifying digitally signed DNS data.

### How do you imagine that a target customer will find this experience?

Cloudflare’s implementation of DNS configuration is simple and intuitive, there is no overcomplication of the deployment nor configuration process.

# Question 3 (Securing Communication)

Secure the communication between Cloudflare and your Origin Server with a non-Cloudflare provisioned TLS certificate using at least the Full-Strict mode on Cloudflare.

## Method 1 (Encrypting origin with Let’s Encrypt)

Earlier certbot was installed in Linux origin, run certbot to request a free certificate from Let’s Encrypt with the command below :

sudo certbot certonly --standalone -d example.techaddressed.com

The certificate will be stored in **/etc/letsencrypt/live/seanproject2024.xyz**, you would need to configure the server block in **/etc/nginx/sites-available/nginx.conf** to point to the certificate when SSL is requested  
A screen shot of a computer

Description automatically generated  
The reason why the picture is not an accurate depiction, is because I have requested Let’s Encrypts API too many times and was rate limited, therefore an alternative is explored

Run sudo /etc/init.d/nginx restart to enable the change

## Method 2 (Encrypting origin with ZeroSSL)

1. Visit <https://zerossl.com/> and register an account to get the free SSL
2. Login and request for a new certificate with the domain  
   A screenshot of a computer

   Description automatically generated
3. Download the certificate, the file will contain 3 items  
   A screenshot of a computer

   Description automatically generated  
   A screenshot of a computer

   Description automatically generated
4. Upload the contents into the origin webserver, I placed it under the **/etc/nginx/** directory for easy access
5. Combine the content of both certificates.crt and ca\_bundle.crt files  
   cat certificate.crt ca\_bundle.crt >> certificate.crt
6. Edit the configuration of **/etc/nginx/sites-available/nginx.conf** with the command **sudo vim /etc/nginx/sites-available/nginx.conf** and edit the server block as suchA screen shot of a computer code

   Description automatically generated
7. Run the command sudo /etc/init.d/nginx restart to enable the change

## Enabling Full-Strict mode on Cloudflare

Under **SSL/TLS > Overview** , choose Full (Strict) to enable end-to-end encryption from client browser to origin server

A screenshot of a computer

Description automatically generated

## What use cases can you see different products being useful for?

Depending on the security requirements and business nature of the customer, the flexibility of customization and implementation of SSL caters to the specific needs of website setups.

For example, if there is an issue with origin certificate, user can opt for flexible or full encryption to secure communication channels between client, Cloudflare proxy, and origin server if there is an issue with the origin’s certificate, ensuring security of a certain degree. It also simplifies since Cloudflare enables free edge certificates with very customizable degrees of settings like TLS version, HTTPS rewrite etc.

## How did you fill the gaps (if any) in your knowledge during the process?

Official Documentation: Referring to official documentation for ZeroSSL API the understanding and implementation of certificate in the origin server.

Discussion forums: Referring to the discussion forum of Let’s Encrypt allowed me to understand the problem of the matter, when certbot failed to call the API, it was due to rate limitations when a certain request for the same domain exceeds a certain amount within a week.

Search Engines and Blogposts: Search engines such as Google enable me to find a free alternative solution to Let’s Encrypt when there is a problem with certificate issue.

## How do you imagine that a target customer will find this experience?

A user-friendly dashboard, making it easy for customers to configure and manage SSL/TLS settings. The intuitive interface contributes to a positive user experience. The experience to configure each security item is also seamless and comprehensive.

# Question 4 (Rate Limiting)

Set a rate limiting rule on Cloudflare. How would you demonstrate to a customer that this rate limiting rule works?

Configurations for rate limiting happens under **Security > WAF**A screenshot of a computer

Description automatically generated

## Examples of configurations with custom rules

To limit access for users from a specific country

A screenshot of a computer

Description automatically generated

To limit access to a request from a specific IP/SubnetA screenshot of a computer

Description automatically generated

## Example of configuration with rate limitations

With the rules configured below, the rule will **block** whatever requests to URL **seanproject2024.xyz/secure** that happened from the **same IP** with the **request count of 1000** within a **10 seconds** periodA screenshot of a computer

Description automatically generated

## What use cases can you see different products being useful for?

Login and Brute force protection: The product can play a role in mitigating brute force attacks and web scraping activities. Such actions require many attempts to login to find the matching credentials intended by the attacker. Such limits for the login page will twat login or data breaches as the attempts allowed is limited.

Resource restriction for specific contents: Rate limiting under WAF enables restricted access to certain resources from certain IP ranges. Exceptions could also be configured to enable a more effective block. Stricter rate limits or conditions can be placed on sensitive contents of a web service that should not be made available to the public.

Bot mitigation: Rate limiting can assist in both bot identification and mitigation. Such bots could exploit vulnerabilities or deny the service of a domain.

API Protection: Rate limiting can protect APIs from abuse, by setting the limits on the number of API requests per client.

## How did you fill the gaps (if any) in your knowledge during the process?

Official Documentations:

<https://developers.cloudflare.com/waf/>

<https://developers.cloudflare.com/ruleset-engine/rules-language/>

## How do you imagine that a target customer will find this experience?

Ease of use: Cloudflare’s Interface for WAF is straightforward, the benefit ease of usability with the defined Boolean and functionality, and it also caters to higher level users with more intricate requirements through configurations with expressions with defined functions that could be referred in Cloudflare’s official documentation.

Flexibility on customization: The rate limiting function provided by Cloudflare enables a high degree of flexibility when it comes to configurations and rule parameters. Security policies with unique characteristics can be put in place to tailor very specific security measures, especially controlling access to customers’ sensitive data.

Performance: Cloudflare’s WAF improves performance of web pages and ensuring availability of their content to end users with the implementation of security, by reducing malicious requests and overloading traffic from suspicious bots or users. The implementation of WAF also does not yield any negative performance impact during or after configuration deployment.

# Question 5 (Cloudflare Tunnel)

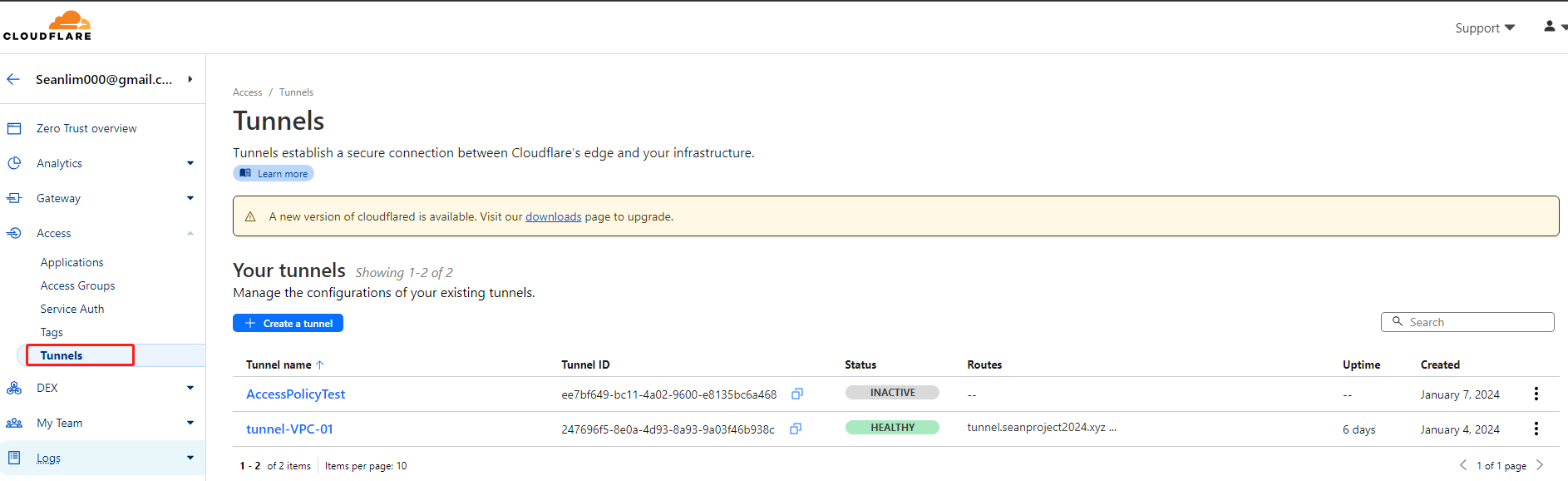
Install and configure Cloudflare Tunnel on your origin server using a subdomain called

“tunnel”, e.g. tunnel.yourwebsite.com. Make connections proxied to your server protected

using this tunnel.

## 

## Implementation

1. Go to Cloudflare’s Launch Zero Trust (one.dash.cloudflare.com)
2. Choose under **Access 🡪 Tunnels**  
   
3. Create a tunnel  
   A white background with black text

   Description automatically generated
4. Name the tunnel and click **Save tunnel**  
   A white background with black text

   Description automatically generated
5. Follow the instructions provided by the page, choose accordingly to the OS of the origin and its appropriate architecture  
   The origin used for the seanproject2024.xyz is Ubuntu LTS, therefore the Debian distribution was used with 64-bit, the commands that are instructed are put into the origin to install cloudflare and run the tunnel. Click **Save** to save the configuration and proceed to the next step.A screenshot of a computer

   Description automatically generated
6. Define the path of the tunnel,   
   Subdomain : tunnel  
   Domain : seanproject2024.xyz  
   Path :  
   Type : http / https (both works, however for HTTPS, **No TLS Verify** should be turned on as the edge certificate and the origin certificate might be different and verification will not pass)  
   URL : Origin IP/hostname  
   A screenshot of a computer

   Description automatically generated
7. You can define the accessible IP blocks of whitelist within the private network tab to limit visits  
   A white rectangle with black lines

   Description automatically generated

## What use cases can you see different products being useful for?

Zero Trust Network Model: Cloudflare tunnel enables a Zero Trust Security Model, that assumes no entity, no matter from an external or internal an organizational network should be trusted and authenticated for access by default. This ensures that accessibility to content is confidential and allows to controlled demographic.

Limit security vulnerabilities from unauthorized device: Cloudflare tunnel can be used to limit accessibility from vulnerable or unsecured devices such as cell phones or IoT devices that could compromise the security of the web application when connected to a certain part of the organization’s resources.

Consistent Security Policy: Cloudflare Tunnel allows for global wide security accompanied with ease of use and configuration. Compared to a list of rules which could present vulnerabilities or loopholes.

Enable security compliance and audit requirements: For organizations with sensitive data at stake, customers will be able to meet regulatory requirements set by their own organizations, as well as third parties’ auditors or government regulations. Cloudflare Tunnel as an industry standard allow auditors to easily audit enforcement in respect with local regulations and company policy.

## How did you fill the gaps (if any) in your knowledge during the process?

YouTube: Watched video introduction on Cloudflare Tunnel from various channels.

Cloudflare’s tunnel configuration page: The page during setup has defined the process of setting up very clearly.

## How do you imagine that a target customer will find this experience?

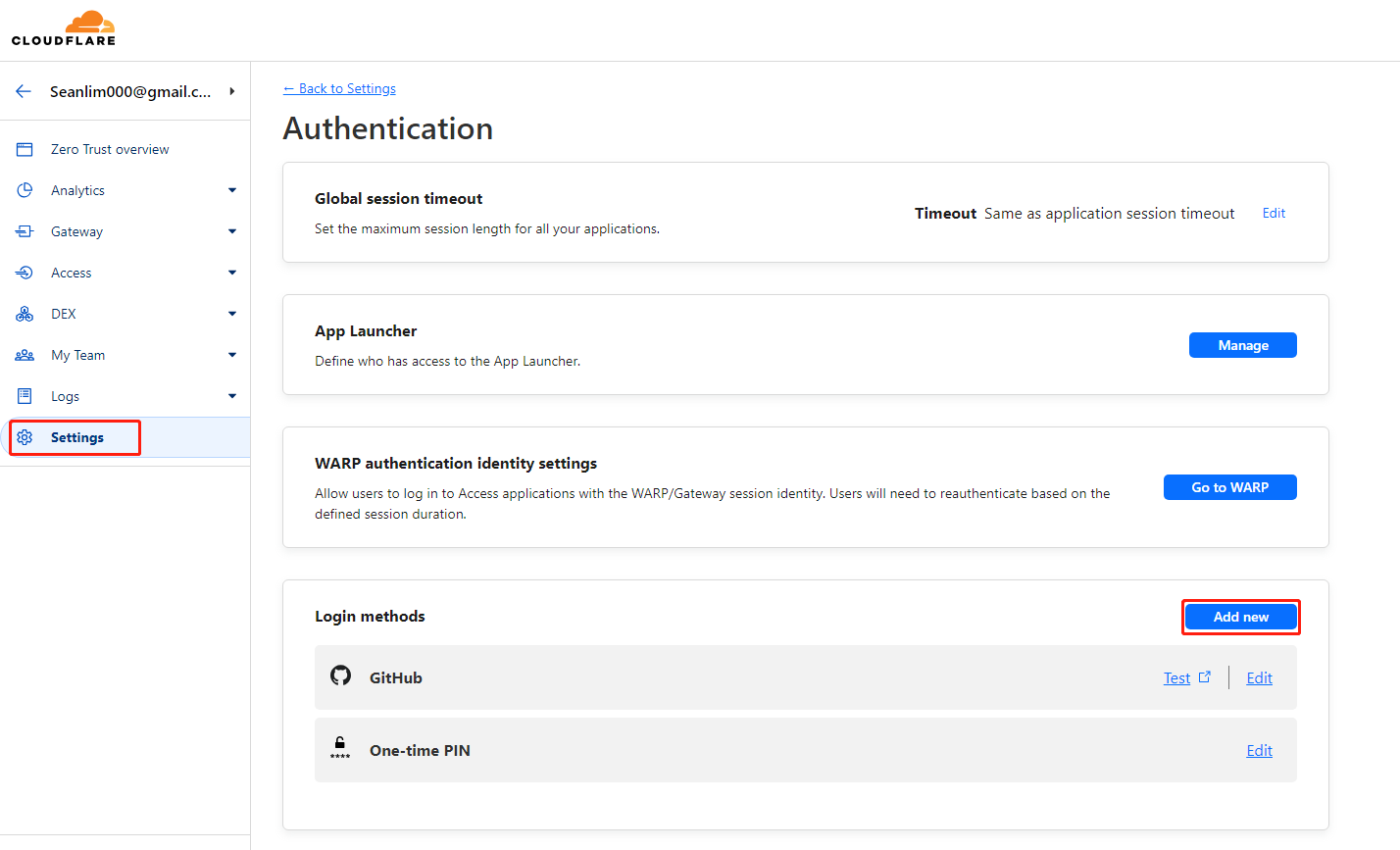
Ease of implementation: The target will be able to implement and configure Cloudflare tunnel with ease, the instruction of use and implementation is clearly defined during the configuration stage.

Confidence: With a Zero Trust security model, customers are confident with the performance, and the security of their web applications. The traffic that is proxied through Cloudflare tunnel can assure confidentiality of customer data and give the customer a peace of mind from attacks or illegal access.

# Question 6 (SSO Idp)

Configure an SSO IdP (Identity Provider) of your choosing within Cloudflare Zero Trust

## Implementation

1. Go to Cloudflare’s Launch Zero Trust (one.dash.cloudflare.com)
2. Go to **Settings > Login methods**
3. Select a login method for SSO  
   A screenshot of a web page

   Description automatically generated  
   In my case, I have used Github as my IdP of choice for Cloudflare SSO
4. Follow the instructions provided on the setup page, log into the GitHub account (https://github.com/login).
5. Under GitHub’s **Settings** > **Developer Settings**, select **OAuth Apps** and select **Register a new application**. The **Register a new OAuth application** window displays.  
   A screenshot of a computer

   Description automatically generated  
   A screenshot of a computer

   Description automatically generated  
   A screenshot of a computer

   Description automatically generated
6. Enter the field of information that is provided in the guide  
   A screenshot of a computer

   Description automatically generated  
   A screenshot of a computer

   Description automatically generated  
   Note: If the team name has been forgotten, you can check it under **Cloudflare’s Launch Zero Trust (one.dash.cloudflare.com) > Settings > Custom Pages**
7. A client ID and Secret Key will be provided after the IdP authentication token is created paste both information in Cloudflare’s SSO pageA screenshot of a computer

   Description automatically generated  
   A screenshot of a login form

   Description automatically generated

## What use cases can you see different products being useful for?

Simple Centralized Login Management: It can function as a simple but complete login management system for employees, partners, or customers based on business function requirements. Resources can be segregated accordingly to appropriate permission control groups for data access.

Reduced IT Overhead: There would not be a need for standalone team to independently function to update, manage or maintain a unique different authorization system to web application contents. SSO in Cloudflare would enable companies to reduce the need for people because of the readily available Login system that is already integrated with Cloudflare, support can be outsourced to Cloudflare’s support team rather than an independent specialized team.

Security for Data Control: With the login management service being integrated and available to the Cloudflare natively, there would be less risk of data breaching or vulnerabilities from malicious party.

## How did you fill the gaps (if any) in your knowledge during the process?

The guide provided in the setup is self-sufficient and direct as a user guide. However an official document is useful (https://developers.cloudflare.com/cloudflare-one/identity/idp-integration/#set-up-idps-in-zero-trust)

## How do you imagine that a target customer will find this experience?

Easier resource management: Makes it easier to allocate web resources to the correct employee. For example, IT team can easily configure and provide access to new onboarding employees, especially if the organization is already using an existing IdP like Okta or Azure AD

User Experience: The user experience while integrating SSO and controlling access for users is easy, employee might also be familiar with the IdP that they have used in their work experience which makes the experience seamless and ubiquitous.

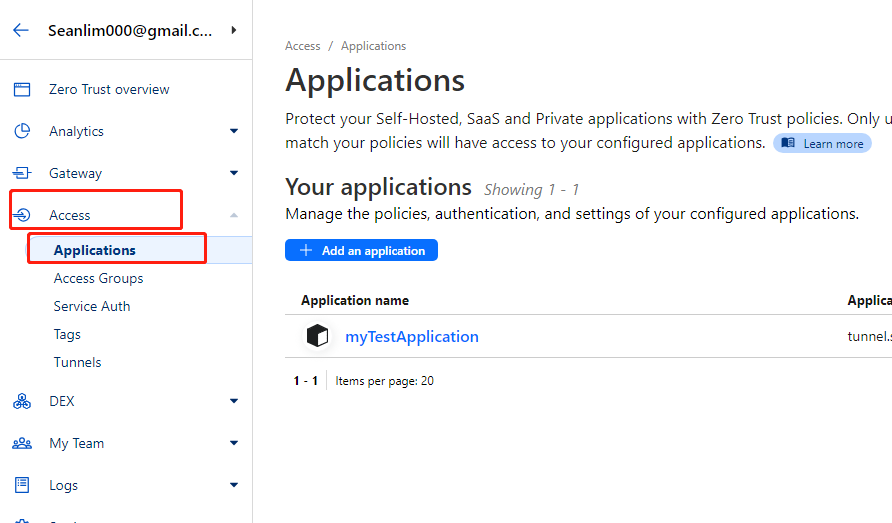
Convenient: SSO eliminates the need for repeated login, improving productivity and reducing the waste of time.

# Question 7 (Lock Down Tunnel Access)

Lock down access for a particular path for your Cloudflare Tunnel subdomain (e.g. tunnel.yourwebsite.com/secure) and only allow access for yourself (with the previously configured IdP of your choosing) and users with an @cloudflare.com email address.

* 1. Ensure nobody can bypass Cloudflare and access your server’s IP directly.

## Implementation

1. Go to Cloudflare’s Launch Zero Trust (one.dash.cloudflare.com) > **Access > Application**
2. Create an application to enable lockdown for a specific path  
   A screenshot of a computer

   Description automatically generated
3. Choose the appropriate application based on the needs  
   A screenshot of a computer

   Description automatically generated  
   **Self-hosted** will be chosen because the requirement was to ensure that the origin server’s IP is not found by using Cloudflare’s authoritative DNS
4. Define the path of the app and name the application. Session duration to expire is set to immediately in order to ease the process of testing instead waiting for long expiry time period  
   A screenshot of a computer

   Description automatically generated
5. Under the **Policies** tab, add a policy to define the control group policy for users  
   A white rectangular object with black lines

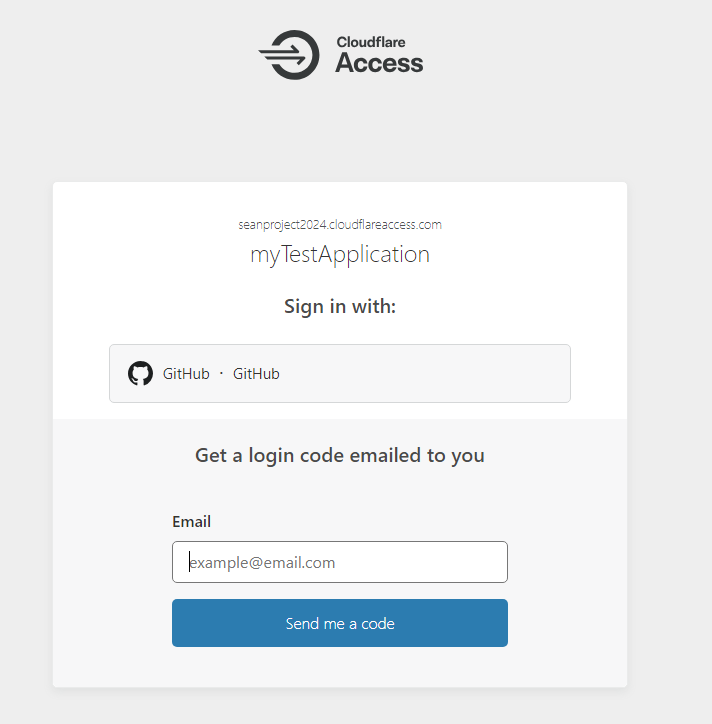
   Description automatically generated
6. There is two way of defining rules for access policy
   1. Method 1: By assigning a already defined group from a access policy that was already defined under **Access > Access Groups**A close-up of a white background

      Description automatically generatedA white background with text

      Description automatically generatedClick **Add a Group** to create a new policy of rules to define access perimA screenshot of a computer

      Description automatically generated  
      Under **Assign a group**, the created policy control group will be listed and you can tick to implement the rules that are defined earlier A screenshot of a computer

      Description automatically generated
   2. Method 2: Create additional rules that are only for the specific application access under **Create additional rules**  
      A screenshot of a computer

      Description automatically generated
7. Click **Save policy** and the changes will take effect  
   

## What use cases can you see different products being useful for?

Securing confidential applications: Locking down applications that are meant for internal within the organization, perhaps due to sensitive data or functionality, Tunnel access lockdown mitigates public accessibility by unauthorized individuals or bots.

Enable Multi-factor authentication (MFA): To prevent unintended accessibility, a MFA is enabled through tunnel access lockdown to ensure that the correct group of people are whitelisted and allowed access. MFA also prevents bots and injections to occur.

## How did you fill the gaps (if any) in your knowledge during the process?

Official documentation: <https://developers.cloudflare.com/cloudflare-one/applications/>

## How do you imagine that a target customer will find this experience?

Reduced complexity: Tunnel lockdown has eliminated the need for complex or expensive solutions, such as on-premises VPN/firewalls, that might need a higher overhead to maintain and scale.

Reduced Downtime: The application is managed and integrated by Cloudflare, therefore there is no potential downtime that could cause security breaches.

Speed of effectiveness: Once the configuration was configured and deployed, the effect was almost instantaneous from the end users’ side. The speed of deployment could reduce the window of vulnerability to attacks.

# Question 8 (Cloudflare Worker)

Serve a Worker on the tunnel.yourwebsite.com/secure path that returns identity information for the authenticated user. The HTTP response body should contain: “${EMAIL} authenticated at ${TIMESTAMP} from ${COUNTRY}” ${COUNTRY} should be an HTML link that when clicked navigates to tunnel.yourwebsite.com/secure/${COUNTRY} and displays the appropriate country flag. The flag asset should be stored in a private R2 bucket.

a. Create this Worker using the Wrangler CLI, upload your Workers code to a public Git repository for your implementation.

b. The /secure response should be returned as HTML

c. The /secure/${COUNTRY} response should use an appropriate content type.

## Implementation

### Sub-Question 1

1. Prepare the pre-requisite.
   1. Sign up for NPM (<https://docs.npmjs.com/creating-a-new-npm-user-account>)
   2. Install nodejs (<https://nodejs.org/en/>)
   3. Visual Code Studio (or any IDE for that matter, since it just makes viewing and correcting codes easier)
   4. Run npm install -g wrangler to install wrangler
2. Run npm create cloudflare@latest to create a new worker project in a terminal window (CMD)
3. Name the new worker directory  
   Select “**Hello World” script**Choose **Yes** to using TypeScript  
   During the script run, you will be required to login with your NPM account that was just registered in the earlier step  
     
     
   A screenshot of a computer program

   Description automatically generated  
   A screenshot of a computer

   Description automatically generated
4. Commands of common use when running wrangler and NPM  
   <https://cloudflareworker-test.seanlim000.workers.dev> 🡨 View deployed application  
   cd cloudflareworker-test 🡨 Navigate to the new directory to edit  
   npm run start 🡨 Run the development local server  
   npm run deploy 🡨 Deploy your application to production  
   <https://developers.cloudflare.com/workers> 🡨 Link to documentation  
   npm run dev 🡨 Deploy your application to local environment for testing  
   <http://127.0.0.1:8787> 🡨 Once the configuration is saved, use local browser to see the effects of change
5. Link to GitHub repository ()

### Sub-Question 2

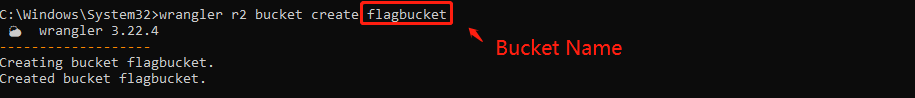
1. Go to cloudflare dash, under **Workers Routes**  
   A screenshot of a computer

   Description automatically generated
2. Click **Add route** , define the path and link the relevant worker with that path  
   A screenshot of a computer

   Description automatically generated

## Sub-Question 3

### Method 1 (That failed)

1. Run wrangler r2 bucket create flagbucket to create a new bucket   
   
2. Run wrangler login to login to wrangler account in browser.
3. Once successful, run wrangler r2 bucket list to ensure that the bucket is created.  
   A black screen with white text

   Description automatically generated
4. Run wrangler whoami to find your account ID, this will be used to bind the bucket in wrangler.toml file.  
   A screenshot of a computer

   Description automatically generated
5. Link the R2 bucket and worker  
   Navigate into the wrangler.toml file, and configure the lines as such :   
   name = worker name  
   main = worker file destination  
   binding = variable key that will later be called to retrieve function from bucket  
   bucket\_name = name of the R2 bucketA black rectangle with white lines

   Description automatically generated  
   You can check if the variable is successfully created by heading onto **Workers&Pages > the worker > Settings > R2 Bucket Bindings**A screenshot of a computer

   Description automatically generated
6. Run npm run deploy to deploy changes to production.

Failed to complete the task beyond this point due to issue of calling “Get” for the appropriate content from R2 Bucket

### Method 2 (Which worked but compromises the requirement)

1. Create a bucket under **R2 > Overview > Create Bucket**A screenshot of a computer

   Description automatically generated
2. Insert bucket name and **Create Bucket**  
   A screenshot of a computer

   Description automatically generated
3. Once created, under **Settings** tab, **Connect Domain** under the **Public Access** setting to enable public access to R2 bucket, this will enable the bucket to be publicly accessible via the domain.  
   A screenshot of a computer

   Description automatically generated  
   A screenshot of a computer

   Description automatically generated  
   Insert the domain name, and the root domain will point to the bucket via CNAME, and the content can be accessed after it is uploaded to the bucket

## What use cases can you see different products being useful for?

Hosting of different content type: R2 can be used to host static, cloud-native applications, web contents, data lakes, archiving data, backup data, and as a cloud storage output for large batch processes.

## How did you fill the gaps (if any) in your knowledge during the process?

Official Documents:   
<https://developers.cloudflare.com/workers>  
<https://developers.cloudflare.com/workers/get-started/guide/>  
<https://developers.cloudflare.com/workers/examples/>  
<https://developers.cloudflare.com/workers/playground/>  
<https://developers.cloudflare.com/workers/examples/geolocation-hello-world/>  
<https://developers.cloudflare.com/r2/api/workers/workers-api-usage/#4-bind-your-bucket-to-a-worker>  
<https://developers.cloudflare.com/r2/api/s3/presigned-urls/>

YouTube:   
<https://www.youtube.com/watch?v=VEATzKttaMI>  
<https://www.youtube.com/watch?v=Q6WTwZI9-Ko>

## How do you imagine that a target customer will find this experience?

Easy Setup: R2 integrates nicely with other Cloudflare products such as Workers and Access, such seamless integration makes it easier for users to setup and configure on the dashboard.

Secure application: Having both R2 and other products integrated into the same ecosystem will reduce the number of potential vulnerabilities in the system. It also ensures that when the bucket and Workers are pulling and pushing data, it would be much more secure as the environment Is maintained and created by an organization (Cloudflare), instead of a integrated project from multiple providers.

Attractive Flexible pricing: After the first 10GB, Cloudflare implement a pay-as-you-go pricing model with competitive rates that suites different business needs, from small to large workloads with no unnecessary costs.

Scalability: R2 scales to accommodate storage needs of various businesses natures, therefore there is no need for customers to provision additional funds for additional infrastructure as their businesses scale.