**Stage 1: Exposing a Web Application to the Internet**

The challenge is to expose a web application to the internet in AWS and choose the most suitable method among API Gateway Rest/HTTP, Load Balancer Application/Network, and Function URL.

Proposed Solutions/Technologies:

API Gateway Rest/HTTP: A managed service for creating and managing APIs, handling HTTP requests, and routing them to backend services.

Load Balancer Application/Network: Distributes incoming traffic across multiple targets, enhancing availability and fault tolerance.

Function URL: Exposing the URL of a serverless function directly to the internet.

Comparison and Use-Case Example: (yellow – advantages, red – disadvantages)

* API Gateway Rest/HTTP: This is a good choice for a web application that needs to be accessible to a wide range of clients, such as mobile apps, web browsers, and other APIs. API Gateway provides several features that make it easy to secure and manage your APIs, such as authentication, authorization, and rate limiting.  
  However, it can be more expensive than other solutions, such as load balancers and function URLs.  
  Example use-case: web application that allows users to create and manage to-do lists. This is important for a to-do list application, as you want to make sure that only authorized users can access the application and that the application is not overloaded by too many requests.
* Load Balancer Application/Network: This is a good choice for a web application that needs to handle a lot of traffic. Load balancers can distribute traffic across multiple servers, which can help to improve performance and reliability.  
   However, it can be more complex to set up and manage than other solutions.  
  Example use-case: web application that is used by a large number of users (e.g: some gov website which handle request by most of the country citizens like “Bituah Leomi”).  
  By using that solution we can make sure that the application is always available and that users do not experience any performance issues.
* Function URL: This is a good choice for a web application that is lightweight and does not need to be accessible to a wide range of clients. Function URLs are simple to create and use, and they can be used to expose Lambda functions as web endpoints.  
  However, function URLs do not provide as many features as API Gateway Rest/HTTP, such as authentication and authorization.  
  Example use-case: web application that is used to process images.  
  You would want to use a function URL to expose this application to the internet, so that you can process images without having to worry about setting up and maintaining a web server. Function URLs are also a good choice for applications that are only used occasionally, as you only pay for the amount of time that your function is running.

**Stage 2: Secure Terminal Access to EC2 Linux Servers** (yellow – advantages, red – disadvantages)

EC2 Linux servers are often exposed to the public internet, which makes them vulnerable to attack. If an attacker can gain access to an EC2 instance, they can steal data, install malware, or disrupt operations.   
We will discuss two solutions for providing secured terminal access to EC2 Linux servers in a private subnet in AWS from personal computers considering those main goals:

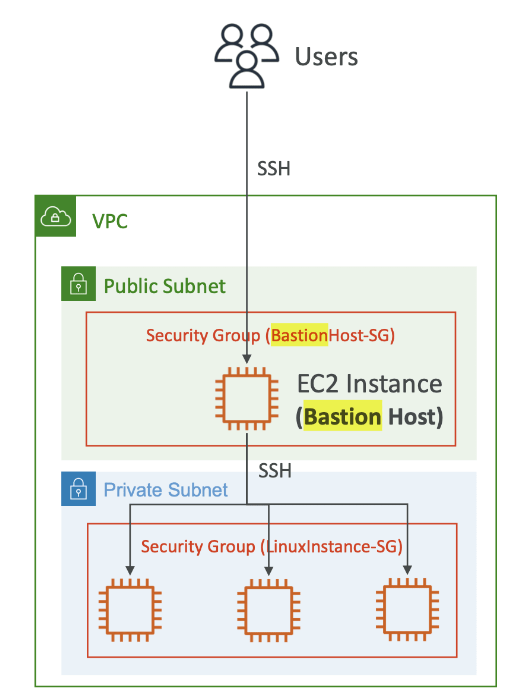
* Security: The solution must be secure, as you will be providing access to the EC2 servers from the internet. You should use strong authentication and encryption to protect the connection.
* Performance: The solution should be performant, as you will be using it to interact with the EC2 servers in real time. You should avoid solutions that add a lot of latency to the connection.
* Ease of use: The solution should be easy to use, as you will need to use it from personal computers that may not be familiar with AWS. You should avoid solutions that require complex configuration or technical expertise.

Here are two solutions that you can use to provide secured terminal access to EC2 Linux servers in a private subnet in AWS from personal computers:

* **Bastion host:** A bastion host is a dedicated server that lets authorized users access a private network from an external network such as the internet. the bastion host becomes the only ingress path to those internal resources. Access control becomes easier to manage while minimizing the potential attack surface  
  a bastion host is an essential component for securing an AWS VPC environment, providing enhanced security, simplified access, scalability, and easy management. However, the bastion acts as Single point of failure - If the bastion host is unavailable, then users will not be able to access the EC2 instances.  
  Management overhead - Bastion hosts require some management overhead. You need to ensure that the bastion host is properly configured and secured.
* **AWS Systems Manager (SSM) and IAM permissions:** AWS Systems Manager works by creating a secure tunnel between your computer and the EC2 instance in the private subnet. It is secured, managed, auditable, IAM integrated and simplified accessed. The tunnel is encrypted, so your traffic is protected from prying eyes. Once the tunnel is established, you can interact with the EC2 instance as if you were connected directly to it.  
  However, can add some latency to the connection between the client and the internal server. This is because the traffic is re-routed through the AWS Systems Manager service.  
  Cost - AWS Systems Manager Session Manager is a paid service. You will need to pay for the number of sessions that you use.

Here is how the proposed solutions work:

* To connect to an EC2 instance using a **bastion host**, you first connect to the bastion host using SSH. Once you are connected to the bastion host, you can then connect to the EC2 instance using SSH. This helps to protect the EC2 instance from direct exposure to the public internet.



* **AWS Systems Manager Session** Manager works by creating a secure tunnel between your computer and the EC2 instance in the private subnet. The tunnel is encrypted, so your traffic is protected from prying eyes. Once the tunnel is established, you can interact with the EC2 instance as if you were connected directly to it.

**A diagram of a software system

Description automatically generated**