Practical Exercise Description:

Availability:
AWS Lambda ensures high availability through its serverless nature, with automatic scaling and operational management. Terraform provides Infrastructure as Code (IaC), enabling quick recreation or repair of the environment during downtime.

Resiliency:
Modular Terraform code and defined outputs allow rapid recovery and efficient redeployment in case of errors.

Recovery:
Terraform's state management and GitHub versioning enable restoring infrastructure to a known good state.

Deliverability:
Updates are automated through GitHub Actions for CI/CD and Terraform for seamless infrastructure changes.

Integrability:
Terraform variables ensure consistent and predictable configurations across environments.

Security:
Access is controlled through IAM roles defined in Terraform. GitHub Secrets and network restrictions enhance security by limiting access to authorized resources only.

Research Exercise:

Bridging DevOps and Security with Scalable Cloud Solutions

Exploring API Gateway, Load Balancers, VPN, ZTNA, Bastion, and SSM in Modern Cloud Architectures

As a DevOps engineer, ensuring scalability, security, and operational efficiency is critical. In AWS, exposing applications to the internet and providing secure terminal access to private resources are foundational challenges. This document explores the tools and technologies that tackle these challenges, their pros and cons, and their relevance to modern DevOps practices.

Stage 1: Exposing Applications to the Internet

The Challenge

DevOps teams frequently need to expose applications to users worldwide while ensuring high availability, performance, and security. In AWS, there are three primary methods to achieve this:

- **API Gateway (REST/HTTP)**
- **Load Balancers (Application/Network)**
- **Function URLs**

Selecting the right tool depends on the application architecture and workload requirements.

Solutions and Comparisons

1. API Gateway (REST/HTTP)

How It Works:
A fully managed service for creating, publishing, and monitoring APIs. It can route requests to AWS Lambda, EC2, or other backend services.

Pros:

- Fine-grained access control with **IAM roles and policies**.
- **Cost-efficient** for serverless applications (HTTP APIs are cheaper than REST APIs).
- Native integration with AWS Lambda, DynamoDB, and other AWS services.

Cons:

- **Latency:** Slightly higher than a direct Load Balancer due to added processing layers.

- **Complexity:** Requires configuration of stages, throttling, and monitoring for APIs.

Use Case:
Ideal for microservices or serverless applications where API-based access is essential. For example, an e-commerce platform using Lambda for backend processing would leverage API Gateway for seamless scalability and low cost.

---

2. Load Balancer (ALB/NLB)

How It Works:
Distributes incoming traffic to multiple targets, such as EC2 instances, containers, or Lambda functions.

- **ALB:** Operates at Layer 7 (HTTP/HTTPS), enabling URL-based routing.
- **NLB:** Operates at Layer 4 (TCP/UDP), ensuring ultra-low latency.

Pros:

- **High Performance:** Suitable for applications with heavy, consistent traffic.
- **Flexibility:** Custom listener rules for traffic routing.
- **Wide Target Compatibility:** Works seamlessly with ECS, EKS, and other AWS compute services.

Cons:

- **Cost:** More expensive than API Gateway for low-traffic applications.
- **Complexity:** Requires additional resources for health checks and scaling.

Use Case:
Best for traditional multi-tier architectures or applications deployed on EC2/ECS/EKS... For example, a web application requiring SSL termination and complex routing would use an ALB.

---

3. Function URLs

How It Works:
Provides a direct HTTPS endpoint for invoking AWS Lambda functions without requiring API Gateway.

Pros:

- **Simplicity:** Quick and easy setup for single-function use cases.
- **Low Cost:** Avoids the overhead of API Gateway for straightforward workflows.

Cons:

- **Limited Features:** No advanced routing, throttling, or caching.

- **Security Constraints:** Basic IAM-based authentication.

  Use Case:
  Suitable for lightweight, single-function applications. For example, a webhook processing Lambda that receives sporadic requests could use a Function URL.

Recommendation

For a **serverless, microservices-based** architecture, **API Gateway HTTP** is the optimal choice due to its cost-efficiency and integration with other AWS services. However, **Load Balancers** excel in high-traffic, EC2-based environments. Function URLs are reserved for simple Lambda-centric workflows.

Stage 2: Secure Terminal Access to EC2 in a Private Subnet

The Challenge

Ensuring secure, scalable, and cost-effective terminal access to EC2 instances in private subnets is a core responsibility in DevOps. Solutions must safeguard the network while enabling productivity.

Solutions and Comparisons

1. VPN (Pritunl)

How It Works:
A VPN server (Pritunl) provides an encrypted connection between the user's device and the AWS private network(VPC).

What to Consider:

- Hosting a VPN server in a public subnet to act as the entry point.
- User authentication, including MFA for enhanced security.

Pros:

- **High Security:** Encrypted communication; optional MFA.
- **Versatile Access:** Supports SSH, HTTP, and other protocols.

Cons:

- **Cost:** Requires an additional EC2 instance and user management.
- **Complex Setup:** Administrators must configure and maintain the VPN server.

Use Case:
Ideal for **multi-user environments** requiring broad, ongoing access to private subnets.

---

2. ZTNA (Zero Trust Network Access)

How It Works:
Access is granted based on identity, device posture, and dynamic security policies.
Example: AWS Verified Access or OpenZiti.
Alternatively, you can opt for a managed service provided by a vendor or integrator to simplify deployment and management, offering expertise and ready-made solutions tailored to your organization's needs.

What to Consider:

- Integration with IAM, device compliance, and continuous monitoring.

Pros:

- **Granular Control:** Access limited to specific resources or applications.
- **Modern Security:** Operates on "never trust, always verify" principles.

Cons:

- **High Cost:** Advanced features increase expenses.
- **Complexity:** Requires robust IAM and device management integration.

Use Case:
Recommended for enterprises with **highly sensitive data** or compliance requirements.

---

3. Session Manager (SSM)

How It Works:
Provides direct CLI access to EC2 instances through the AWS Management Console or CLI, without requiring a public IP.

What to Consider:

- Attaching IAM roles with SSM permissions to the EC2 instances.

Pros:

- **No Additional Cost:** Included in AWS service pricing.
- **Ease of Use:** Requires no additional infrastructure.
- **High Security:** IAM-managed access without open ports.

Cons:

- **Limited to CLI:** Not suitable for graphical applications.

Use Case:
Optimal for **small-scale setups** with limited users needing occasional CLI access.

---

4. Bastion Host

How It Works:
A dedicated EC2 instance in a public subnet acts as an intermediary for accessing private subnet resources.

What to Consider:

- Hardening the Bastion host with strict security policies.

Pros:

- **Direct Control:** Full flexibility over SSH and network access.

Cons:

- **Outdated:** Modern solutions like SSM or VPN provide better solution.
- **High Maintenance:** Requires regular updates and SSH key management.

Use Case:
Considered a **legacy solution** suitable only when other options are unavailable.

Recommendation

- **Session Manager** is the go-to solution for most **modern DevOps teams**, offering simplicity, security, and zero infrastructure cost.
- **VPN** is recommended for environments with **multiple users** and complex workflows.
- **ZTNA** is suited for organizations prioritizing **granular security** and compliance.
- **Bastion Host** should be avoided unless necessary for **legacy applications**.

Conclusion: DevOps-Centric Insights

The chosen technologies reflect core DevOps principles: **automation, scalability, and security**. While **API Gateway and SSM** align with the shift to serverless and cloud-native architectures, **VPN and ZTNA** represent robust solutions for environments needing versatile access controls. Selecting the right approach ensures seamless operations and positions teams to innovate confidently in the cloud.