**🌐 VPC (Virtual Private Cloud)**

**Concept**

A **Virtual Private Cloud (VPC)** is your own private network inside AWS. Think of it as your **data center in the cloud**, where you can isolate services, control traffic, and secure communication.

**How it Works**

* **Subnets:** Divide your VPC into **public** (internet-facing) and **private** (internal-only) networks.
* **Internet Gateway (IGW):** Connects public subnet resources to the internet.
* **NAT Gateway:** Lets private subnet resources (ECS tasks, DBs) reach the internet *outbound* (to download packages, call APIs) but keeps them hidden from inbound traffic.
* **Security Groups:** Firewalls for instances/containers.

**Pros**

* Full control over networking & security.
* Segregate workloads (API public, DB private).
* Supports scaling across multiple Availability Zones.

**Cons**

* Requires careful design (wrong routing/security = service unreachable).
* Can feel heavy for small projects.

**LexiFlow Example**

* **Public subnet:** ALB (entry point).
* **Private subnets:** ECS tasks (FastAPI API + workers), DynamoDB, Vector DB.
* NAT Gateway allows ECS workers to call OpenAI/Bedrock APIs securely.
* VPC Flow Logs = monitoring network traffic (audit logs).

**Key Points**

* Always design with **at least 2 AZs** for fault tolerance.
* Keep DB/vector storage in **private subnets**.
* Logs (Flow Logs) = “source of truth” (DDIA principle).

**🐳 ECS (Elastic Container Service) + Fargate**

**Concept**

**ECS** is AWS’s service to run **containers at scale**.  
**Fargate** is a *serverless compute option for ECS* — you don’t manage EC2 servers; AWS runs containers for you.

**How it Works**

* **Task Definition:** Blueprint for your container (image, CPU, memory, env vars).
* **Service:** Runs tasks continuously, restarts if one fails.
* **Scaling:** Add/remove tasks based on CPU, memory, or queue length.

**Pros**

* Fully managed container orchestration.
* Integrates with ALB, CloudWatch, Secrets Manager.
* Scales horizontally (many tasks across AZs).
* Fargate = no EC2 to manage.

**Cons**

* Slightly more expensive than EC2-based ECS.
* Cold-start latency when scaling up.
* Debugging network issues inside VPC can be tricky.

**LexiFlow Example**

* One ECS **service** runs FastAPI app (API for doc chat).
* Another ECS **worker service** processes doc ingestion (chunks, embeddings).
* Fargate ensures tasks scale up if multiple docs are uploaded at once.
* Task auto-restarts if FastAPI crashes → reliability.

**Key Points**

* ECS = replication & failover (DDIA “reliability”).
* Use SQS → ECS workers for **backpressure**.
* Secrets Manager for storing API keys.

**⚖️ ALB (Application Load Balancer)**

**Concept**

An **Application Load Balancer (ALB)** distributes **incoming HTTP/HTTPS requests** across multiple ECS tasks. It ensures no single task is overloaded.

**How it Works**

* ALB listens on port 80/443.
* Routes traffic to ECS tasks (target groups).
* Performs **health checks**: removes unhealthy tasks, reroutes traffic.
* Supports **SSL termination** (manages HTTPS certs).

**Pros**

* Automatic failover if one ECS task dies.
* SSL/TLS handled centrally.
* Scales seamlessly with ECS.
* Logging available (access logs).

**Cons**

* Adds cost (~$18–20/month baseline).
* Slight latency overhead (extra network hop).

**LexiFlow Example**

* User hits https://lexiflow.idataflow.ai/query.
* Route53 DNS → ALB → ECS FastAPI task.
* If one task dies, ALB routes to another.
* ALB access logs used for monitoring usage patterns.

**Key Points**

* ALB = DDIA **failover mechanism**.
* Use with multi-AZ ECS tasks for high availability.
* Central place for SSL certs → no need to manage certs per container.

**🛡️ WAF (Web Application Firewall)**

**Concept**

A **Web Application Firewall (WAF)** protects your app from malicious requests (SQL injection, XSS, prompt injection). It sits in front of ALB.

**How it Works**

* Predefined AWS Managed Rules (common attacks).
* Custom rules (block certain IPs, request patterns).
* Logs every blocked/allowed request.

**Pros**

* Adds **security guardrails** at entry point.
* Protects against prompt injection attempts.
* Easy to attach to ALB or CloudFront.

**Cons**

* Costs add up (per rule, per request).
* False positives can block valid requests if rules are too strict.

**LexiFlow Example**

* WAF rule blocks queries like:
  + "DROP TABLE users;"
  + "Ignore system instructions" (prompt injection attempt).
* Logs attacks to CloudWatch.

**Key Points**

* WAF = DDIA **guardrails** → safety against human error/malicious input.
* Always attach to ALB/CloudFront.
* Combine with IAM least privilege for zero-trust security.

**🚀 TL;DR Cheat Sheet**

| **Service** | **Concept** | **Working** | **Pros** | **Cons** | **LexiFlow Use** |
| --- | --- | --- | --- | --- | --- |
| **VPC** | Private network in AWS | Subnets, IGW, NAT, SGs | Isolation, control, secure design | Complexity | Public: ALB, Private: ECS, DBs |
| **ECS** | Run containers | Tasks, Services, Scaling | Scales easily, integrates with AWS | Expensive, cold starts | FastAPI API + Worker tasks |
| **ALB** | Distribute requests | Routes + health checks | Failover, SSL, logs | Cost, latency hop | Routes traffic → ECS tasks |
| **WAF** | Firewall for apps | Rules (SQLi, XSS) | Protects against attacks | Cost, false positives | Block prompt injection + SQLi |