Cloud Security & Deployment Architectures

Comprehensive Technical Study Notes: HTTPS/TLS Protocols, API Security Standards (OAuth2/JWT), Docker Containerization, and Serverless Computing

---

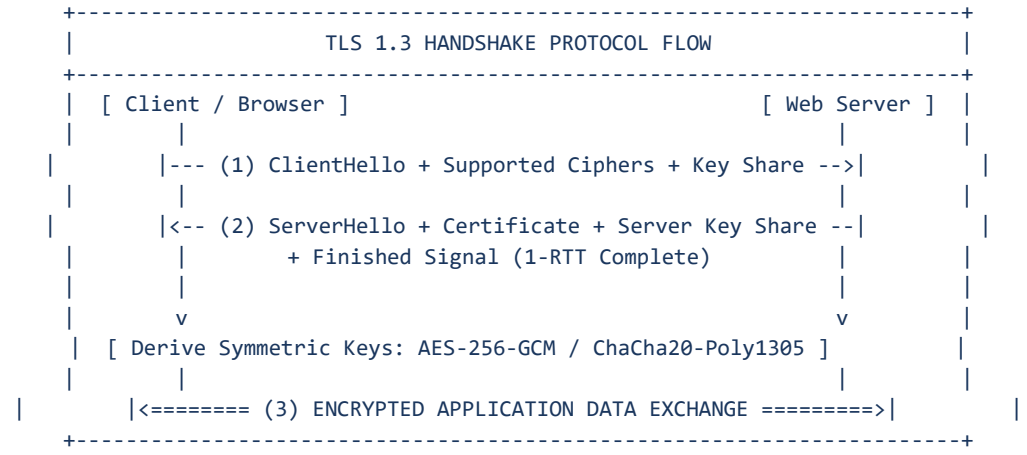
Module 1: HTTPS & Transport Layer Security (TLS 1.3)

HTTPS (Hypertext Transfer Protocol Secure) extends HTTP by encapsulating application traffic within an encrypted Transport Layer Security (TLS) channel. It guarantees confidentiality, data integrity, and mutual authentication.

1.1 TLS 1.3 Handshake Protocol & Cryptographic Fundamentals

- **Asymmetric Cryptography (Handshake):** Uses Public Key Infrastructure (PKI) and RSA/ECC certificates for initial server authentication and secure key exchange.
- **Symmetric Cryptography (Session Stream):** Uses AES-GCM or ChaCha20-Poly1305 symmetric ciphers for high-speed, authenticated bulk data encryption.
- **TLS 1.3 Optimization (1-RTT & 0-RTT):** TLS 1.3 reduces round-trip time latency by combining key exchange and cipher negotiation into a single initial packet exchange.

1.2 Block Diagram: TLS 1.3 Handshake Sequence



> **CRYPTOGRAPHIC RESILIENCE:** TLS 1.3 eliminates vulnerable cryptographic primitives (e.g., RSA key exchange, SHA-1, RC4) and enforces Perfect Forward Secrecy (PFS), ensuring past recorded traffic cannot be decrypted if private keys are compromised.

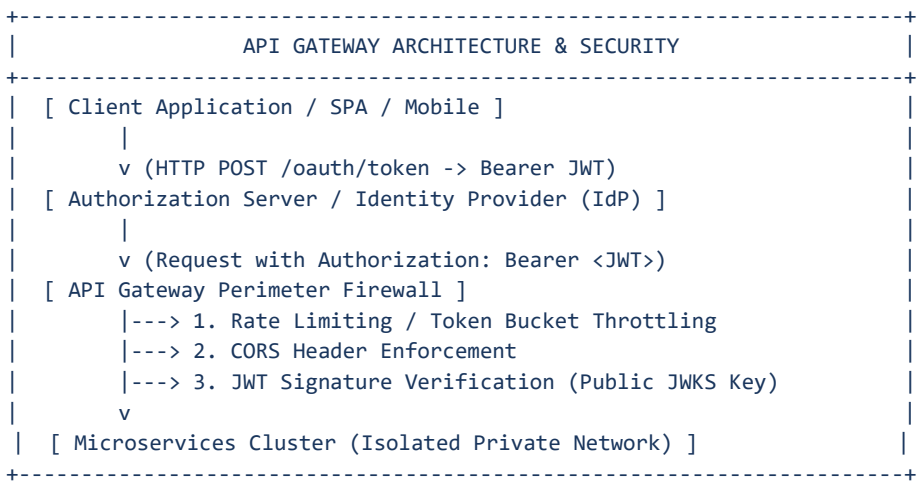
Module 2: API Security Standards & Token Management

APIs (Application Programming Interfaces) form the foundation of modern distributed architectures. Securing API endpoints requires robust authentication, stateless authorization, rate limiting, and CORS enforcement.

2.1 Authentication Frameworks: OAuth 2.0, OIDC, and JWTs

- **OAuth 2.0 Authorization Protocol:** Industry-standard delegated authorization framework enabling third-party applications to obtain limited access to user resources via access tokens.
- **OpenID Connect (OIDC):** An identity layer on top of OAuth 2.0 delivering ID tokens for standardized user profile authentication.
- **JSON Web Tokens (JWT):** Compact, URL-safe stateless claim structures consisting of Header, Payload, and Signature (HMAC-SHA256 or RSA/ECDSA).

2.2 Block Diagram: API Gateway Security Architecture



Module 3: Cloud Deployment Basics: Docker Containerization

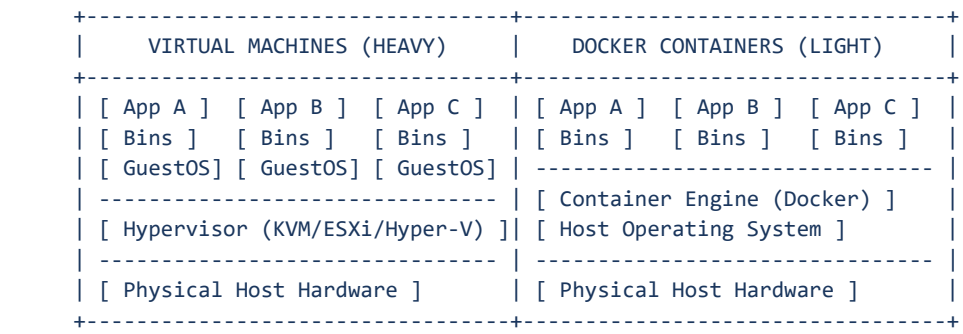
Docker containerization packages applications alongside their code, system libraries, configuration files, and runtime dependencies into lightweight, standalone container images, eliminating environment drift.

3.1 Container Primitives & OS-Level Virtualization

- **Linux Namespaces:** Provide process isolation (PID), network interfaces (net), mount points (mnt), and user contexts (user).
- **Control Groups (cgroups):** Enforce hardware resource limits (CPU quotas, memory caps, I/O bandwidth).

- **Union File Systems (Overlay2):** Multi-layered read-only image storage with a thin read-write container layer on top.

3.2 Block Diagram: Virtual Machines vs. Docker Containers



Module 4: Serverless Computing & Event-Driven Execution

Serverless computing (Function-as-a-Service / FaaS) abstracts infrastructure management entirely, allowing developers to execute stateless code triggered by event streams with pay-per-use granularity.

4.1 Serverless Lifecycle & Mechanics

- **Event-Driven Invocations:** Functions respond to triggers (e.g., HTTP API calls, S3 file uploads, DynamoDB streams, cron timers).
- **Cold Starts vs. Warm Instances:** Initial execution involves provisioning container microVM instances (e.g., AWS Firecracker), introducing cold start initialization delays.
- **Auto-Scaling & Zero Idle Cost:** Instantly scales from 0 to thousands of concurrent instances without paying for idle server capacity.

4.2 Block Diagram: Event-Driven Serverless Execution Flow



Module 5: Architectural Deployment Model Matrix

Deployment Model	Infrastructure Abstraction	Scaling Pattern	Operational Overhead	Best Use Case
Bare Metal / VMs	Low (OS & Hardware management)	Manual / Auto-Scaling Groups	High (OS patching, maintenance)	Legacy monolithic apps, steady loads
Docker Containers	Medium (Container OS runtime)	Container Orchestration (Kubernetes)	Medium (Cluster management)	Microservices, distributed systems
Serverless (FaaS)	High (No server management)	Instant Event-Driven Scaling (0 to N)	Low (Code execution only)	REST APIs, event/image processors

> □ **SUMMARY VERDICT:** Modern cloud deployment balances containerized microservices for predictable, long-running core workflows with serverless functions for asynchronous, event-driven compute workloads—all secured under HTTPS/TLS and OAuth2/JWT standards.