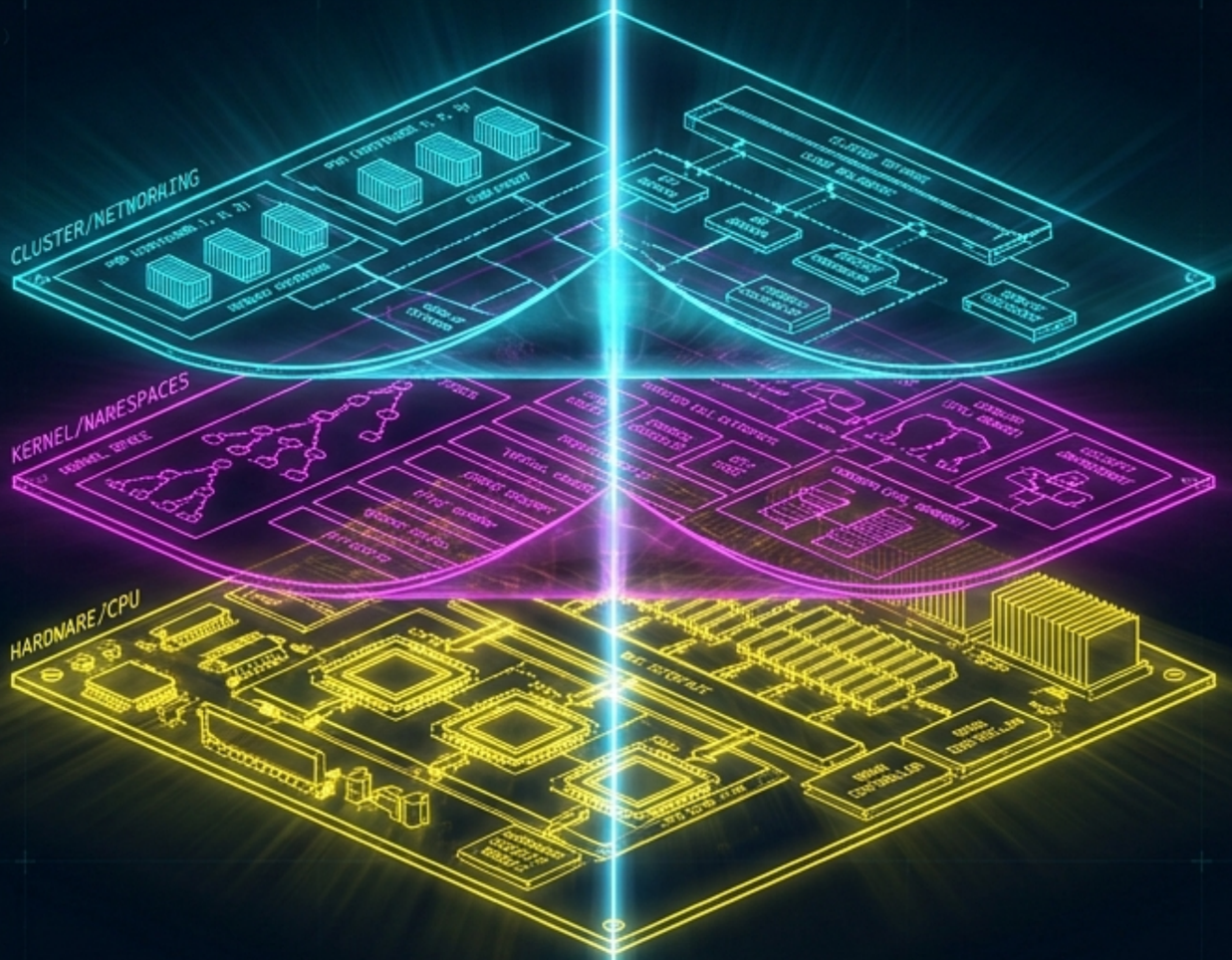
SCALE: 1:1 kk/cgroup.c

STATUS: ACTIVE BLUEPRINT

WEB ARCHITECTURE S02E05

LINUX KERNEL CONTAINER PRIMITIVES, OCI RUNTIME SPECS & KUBERNETES ARCHITECTURE



SCALE: 1:1 VIEW: ISOMETRIC CROSS-SECTION

STATUS: ACTIVE BLUEPRINT

Gemini Notebook

7. **The Reality**
(Production Failures)

6. **The Nervous System**
(K8s Networking)

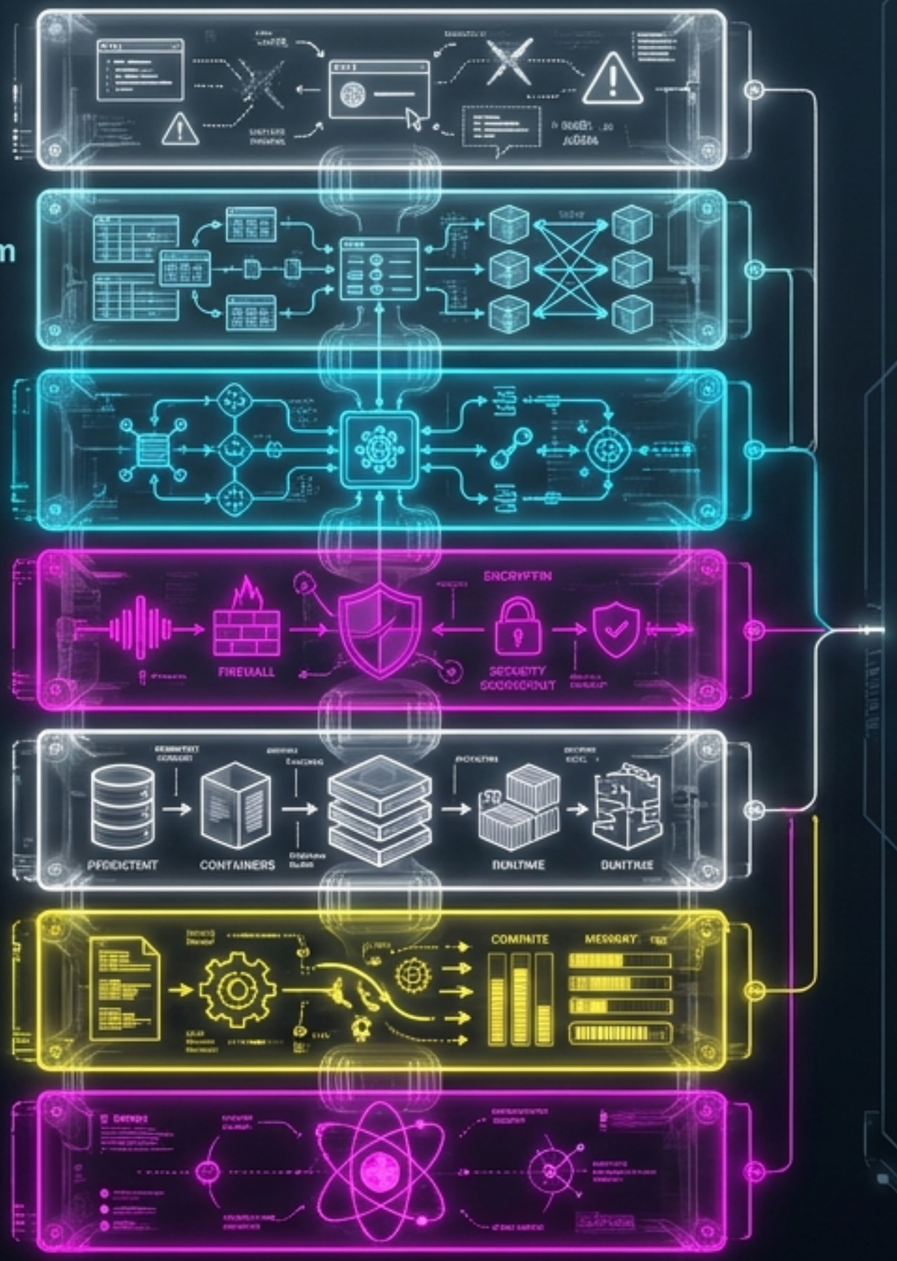
5. **The Brain**
(Control Plane)

4. **The Perimeter**
(Security Spectrum)

3. **The Construct**
(Storage & Runtimes)

2. **The Resources**
(Kernel Governance)

1. **The Atom**
(Kernel Isolation)



The Full-Stack Illusion

A container is an illusion. It is a carefully orchestrated lie told by the Linux Kernel to a set of processes.

This document deconstructs that lie, layer by layer, until it forms a Kubernetes cluster.

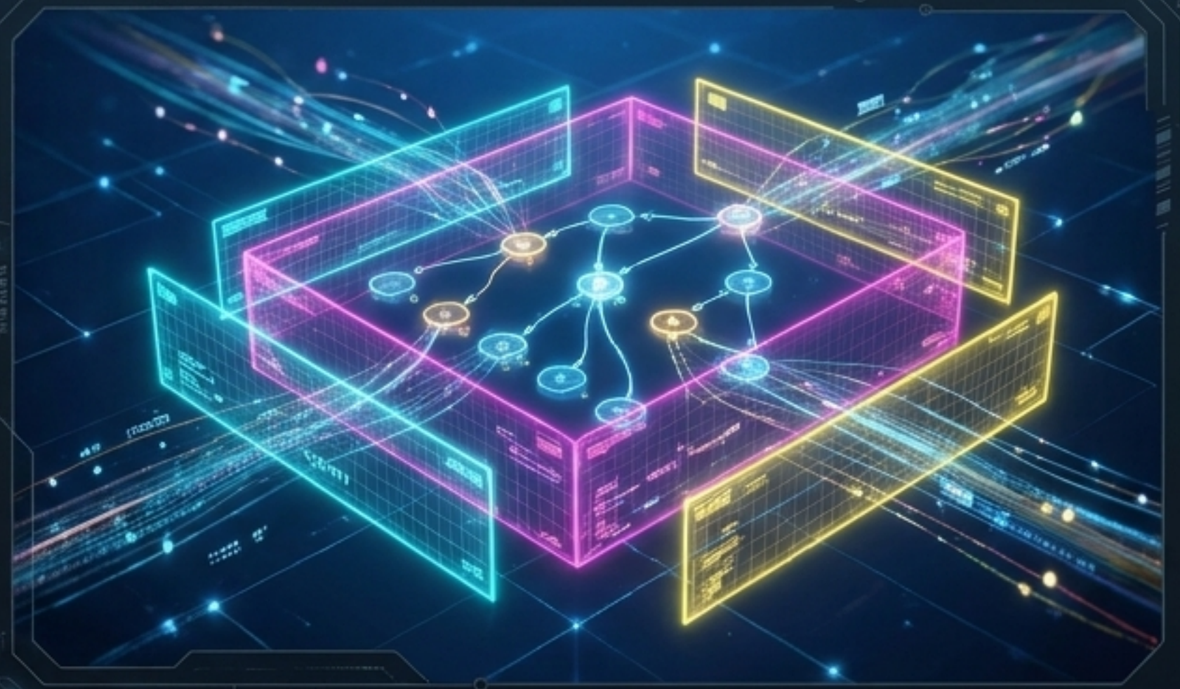
Gemini Notebook

Deconstructing the Container Illusion

The False Model



The True Model

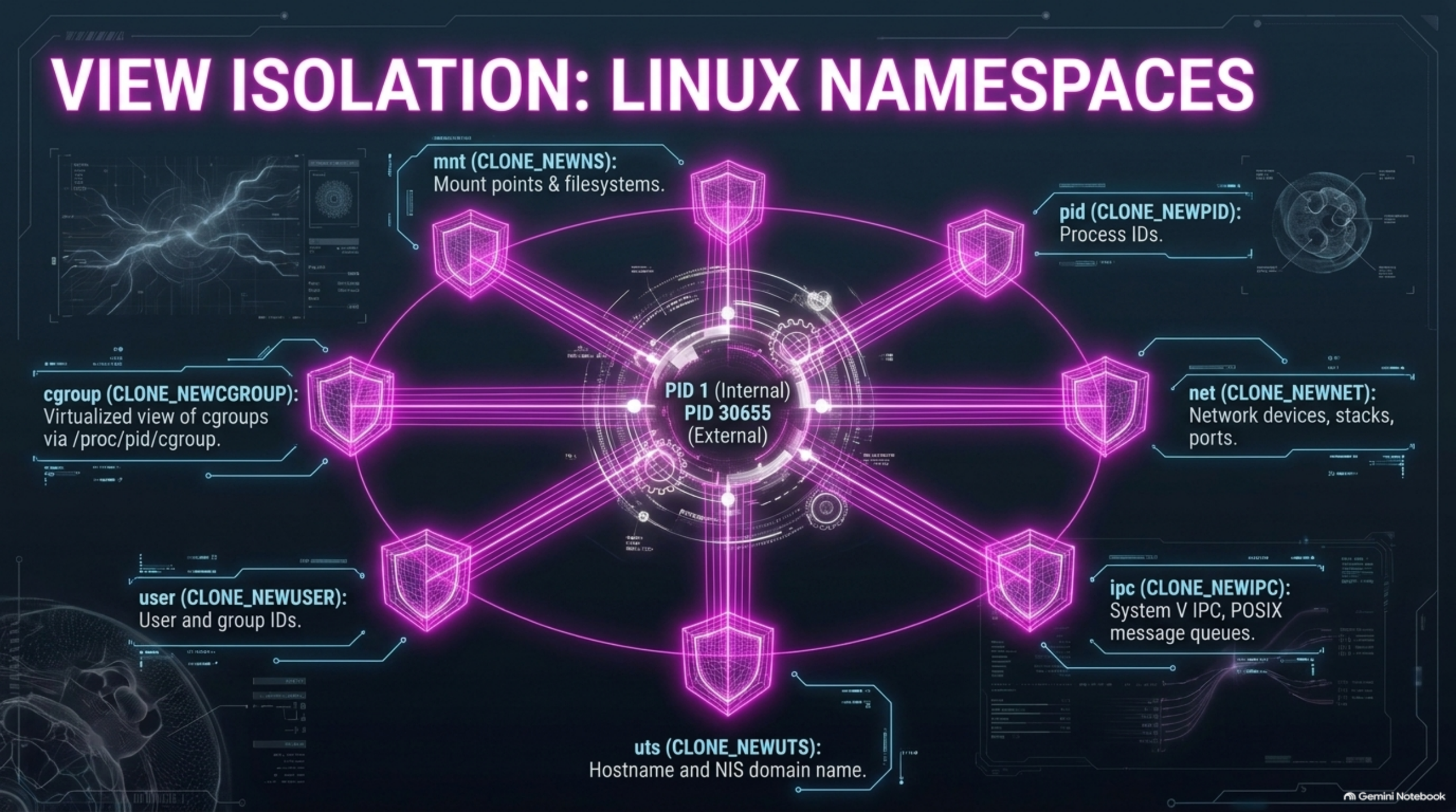


```
int clone(int (*fn)(void *), void *stack, int flags, void *arg);
unshare(CLONE_NEWPID | CLONE_NEWNET | CLONE_NEWNS);
```

- Linux has no `struct container`.
- It relies on `clone()` and `unshare()` manipulating bitmask flags to carve out isolated views of global resources.
- A container is just a normal Linux process with highly restricted visibility.

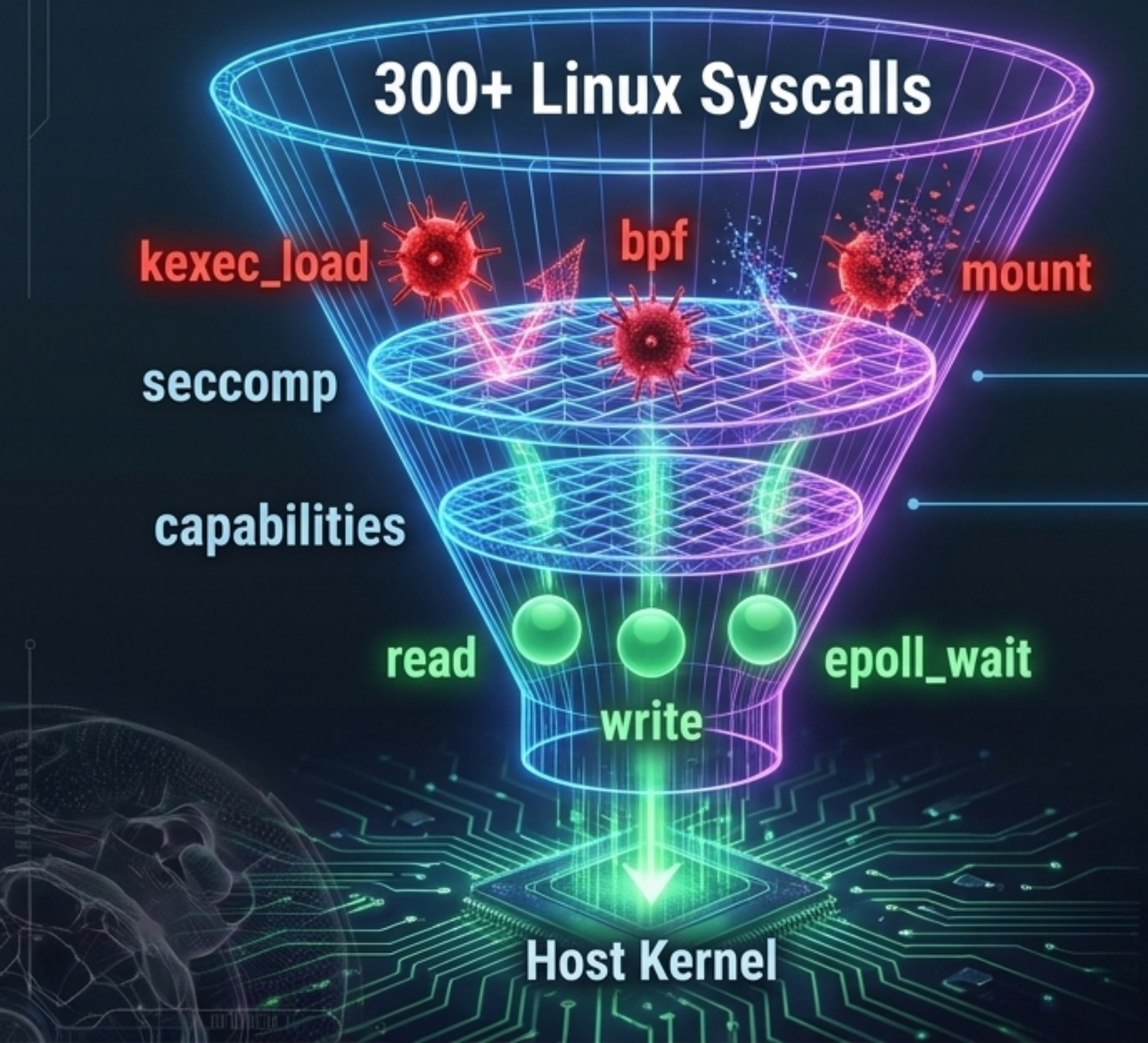
Gemini Notebook

VIEW ISOLATION: LINUX NAMESPACES



Gemini Notebook

Syscall Lockdown: seccomp-bpf & Capabilities



Linux Capabilities

Breaks root privileges into 40+ distinct units (e.g., CAP_NET_ADMIN, CAP_SYS_ADMIN).

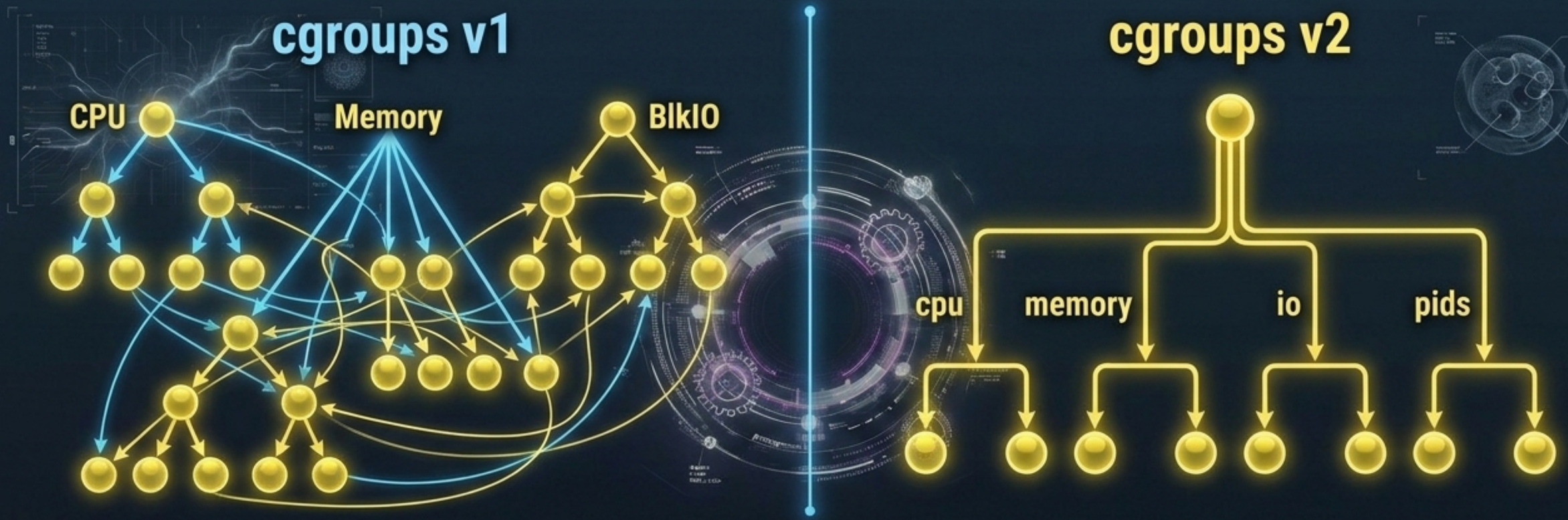
Docker and K8s drop most of these by default.

Seccomp-bpf

Secure Computing mode with Berkeley Packet Filters. A state machine that intercepts syscalls before they hit the kernel. Returns SECCOMP_RET_KILL for forbidden calls.

Gemini Notebook

RESOURCE GOVERNANCE: CGROUPS V2



The Unified Hierarchy

All resource controllers (cpu, memory, io, pids) are mounted against a single unified hierarchy.

No Internal Processes Rule

Processes may reside only in leaf nodes. This avoids resource distribution ambiguity between parent and child processes.

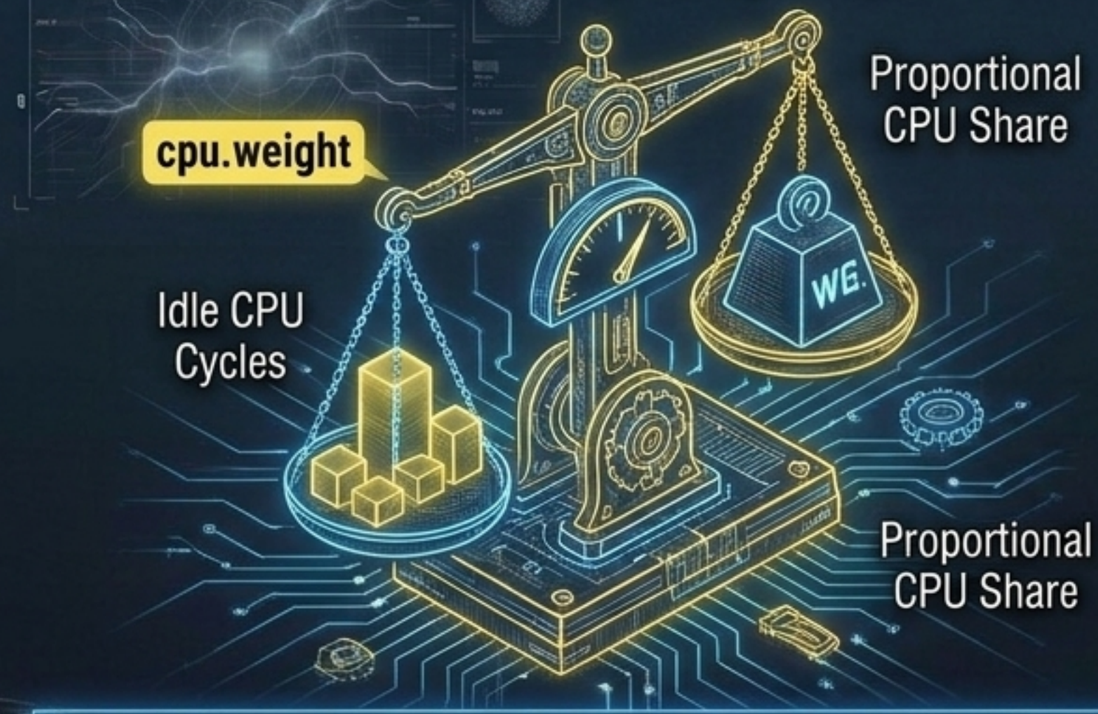
Subtree Control

Active controllers are explicitly delegated down the tree via the cgroup.subtree_control file.

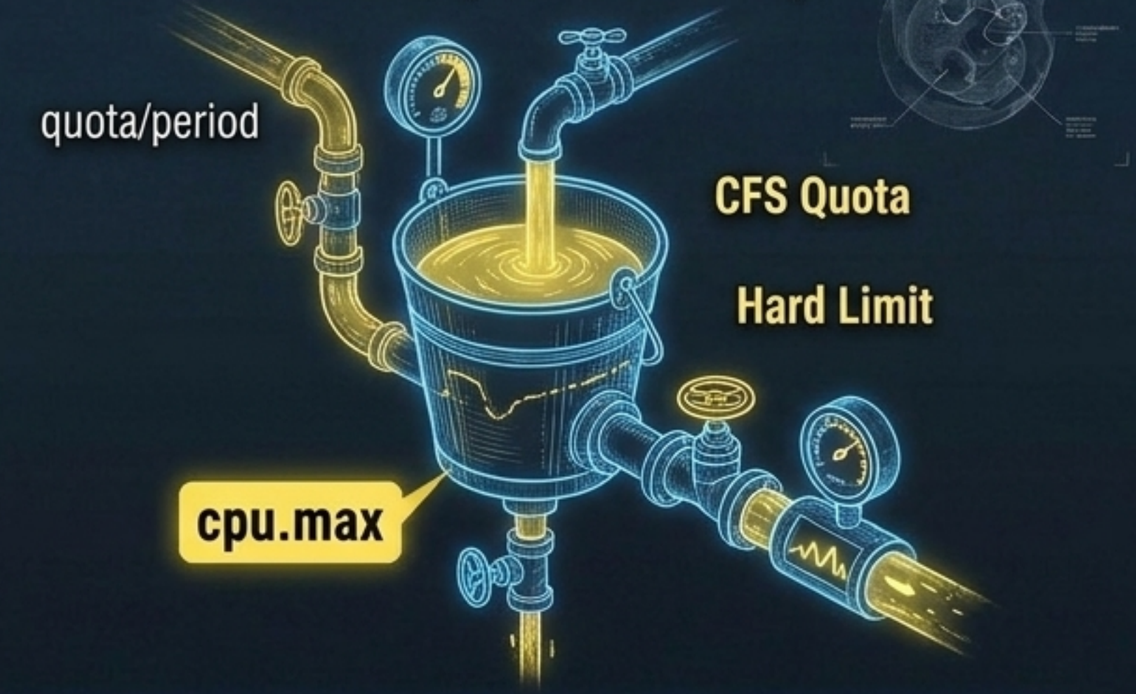
Gemini Notebook

CPU Scheduling: Requests vs. Limits

Requests (Weights)



Limits (Bandwidth)



Requests (Weights): Enforced via `cpu.weight`. Guarantees a minimum proportional CPU share under node contention.

Limits (Bandwidth): Enforced via `cpu.max` (quota/period).

The Multi-Thread Trap | Aggressive limits cause artificial throttling spikes. A highly threaded app can consume its entire 100ms quota in 10ms across 10 cores, pausing completely and freezing for the remaining 90ms.

Gemini Notebook

MEMORY LIMITS & THE OOM KILLER



Enforcement: Strict threshold handled via memory.max.

Incompressible Resource: Unlike CPU, memory cannot be artificially throttled. Hitting the limit leaves the kernel only one choice: termination.

The Executioner: The kernel oom_killer algorithm selects the process and issues SIGKILL (Signal 9). 128 + 9 = Exit 137.

No Grace Period: Applications cannot catch, intercept, or ignore SIGKILL. Data in flight is instantly lost.

Gemini Notebook

OCI Image Specifications



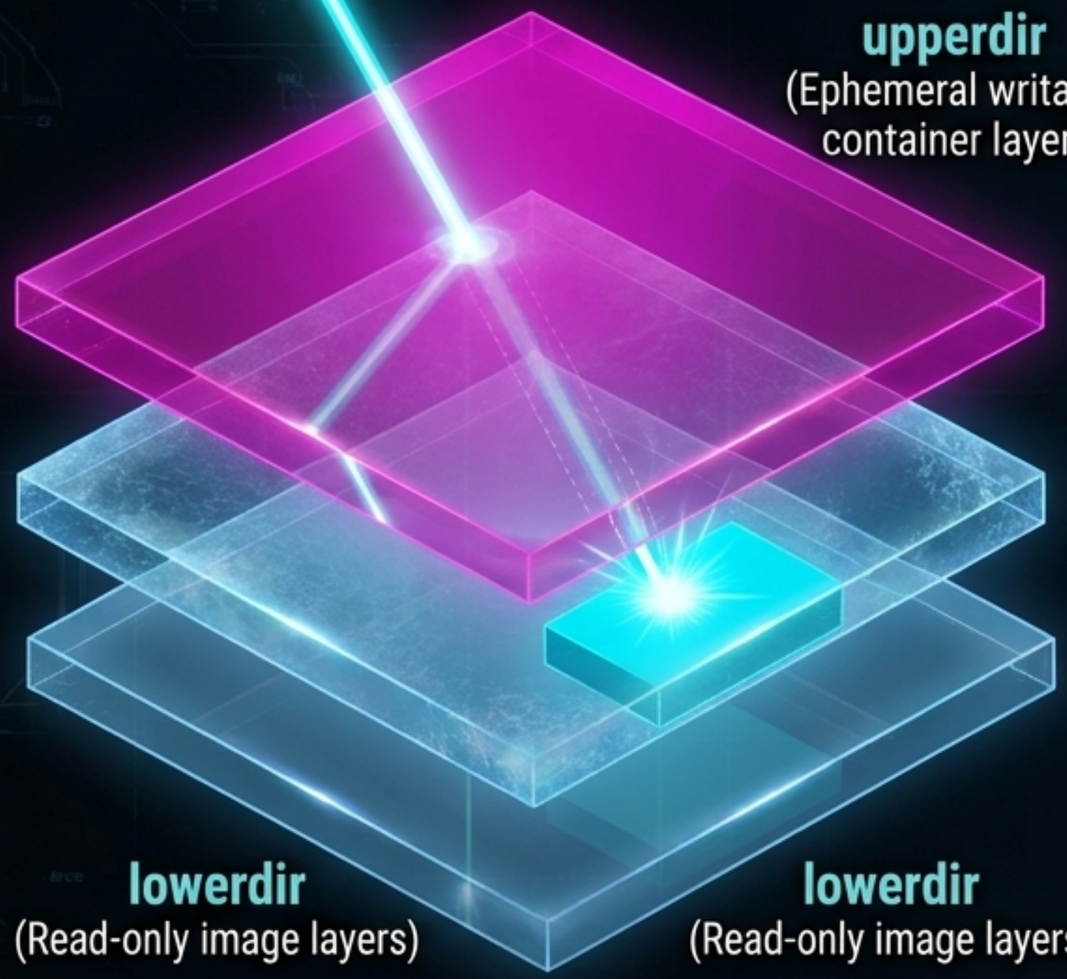
Content-Addressable Tar Layers: Images are not single executable files. They are a manifest of distinct, unpackable file layers.

SHA-256 Identification: Every layer is uniquely identified by the cryptographic hash of its exact contents, not by its name or tag.

Cryptographic Immutability: Changing a single byte in the base layer completely alters the hashes of all subsequent dependent layers.

Gemini Notebook

OverlayFS Architecture



Shared Page Cache: Immutability allows multiple distinct containers to share the exact same lowerdir base image directly in host RAM, radically reducing memory overhead.

Copy-on-Write (CoW) Penalty: Modifying a file that exists in a lower layer forces the kernel to copy the entire file up to the upperdir before saving the changes. Modifying large log files or databases inside the container incurs massive I/O penalties.

Gemini Notebook

The Container Execution Chain



Kubelet
The K8s node agent. It dictates state but does not actually know how to run containers.

CRI Boundary
The Container Runtime Interface is the gRPC boundary decoupling K8s from specific low-level runtimes.

containerd
The high-level runtime. Manages network attachments, pulls images, and handles storage.

runc
The OCI low-level runtime. Parses JSON configs, issues the actual clone() and unshare() syscalls to the kernel, and hands over execution.

Gemini Notebook

Security Isolation Spectrum Matrix

	Standard Container	Virtual Machine	MicroVM (Firecracker)	Sandbox (gVisor)
Kernel Boundary	Shared Host	Dedicated Guest	Dedicated Guest	User-Space
Cold Start Latency	~50ms	~30s	~150ms	~200ms
Syscall Attack Surface	High ⚠	Low	Low	Low
Resource Overhead	Minimal	High	Low	Medium

Takeaway: Shared host kernels are fundamentally vulnerable to zero-day exploits. True multi-tenancy for untrusted code requires escaping the shared host kernel entirely.

Gemini Notebook

Untrusted Multi-Tenant Workloads

AWS Firecracker (MicroVMs)



Uses hardware-assisted KVM isolation. Strips out legacy device drivers (no USB, no floppy) to achieve 150ms boot times. Used by AWS Lambda.

Google gVisor (User-Space Kernel)

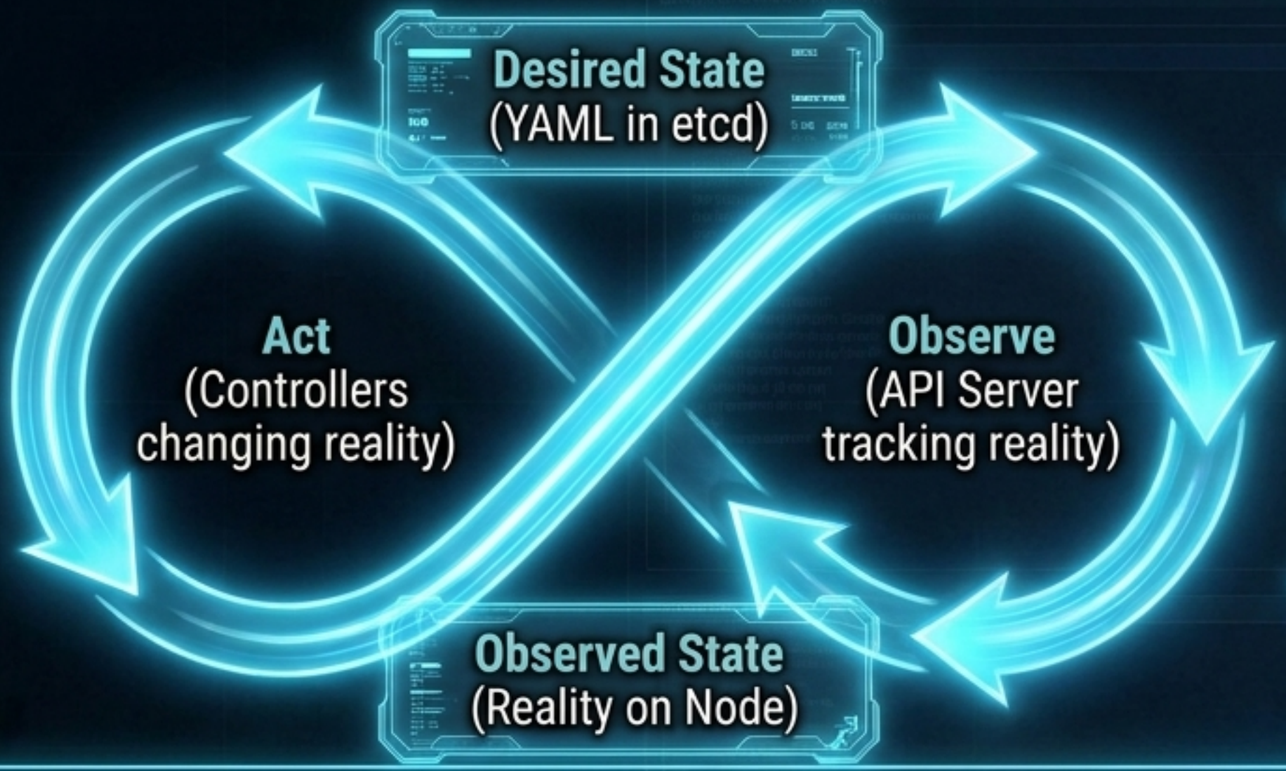


Implements Linux networking and filesystem entirely in memory via Go. Intercepts, sanitizes, and implements all syscalls. Used by Google Cloud Run.

Gemini Notebook

Control-Plane: Level-Triggered Reconciliation

Kubernetes is not an imperative orchestration engine (First do A, then B).

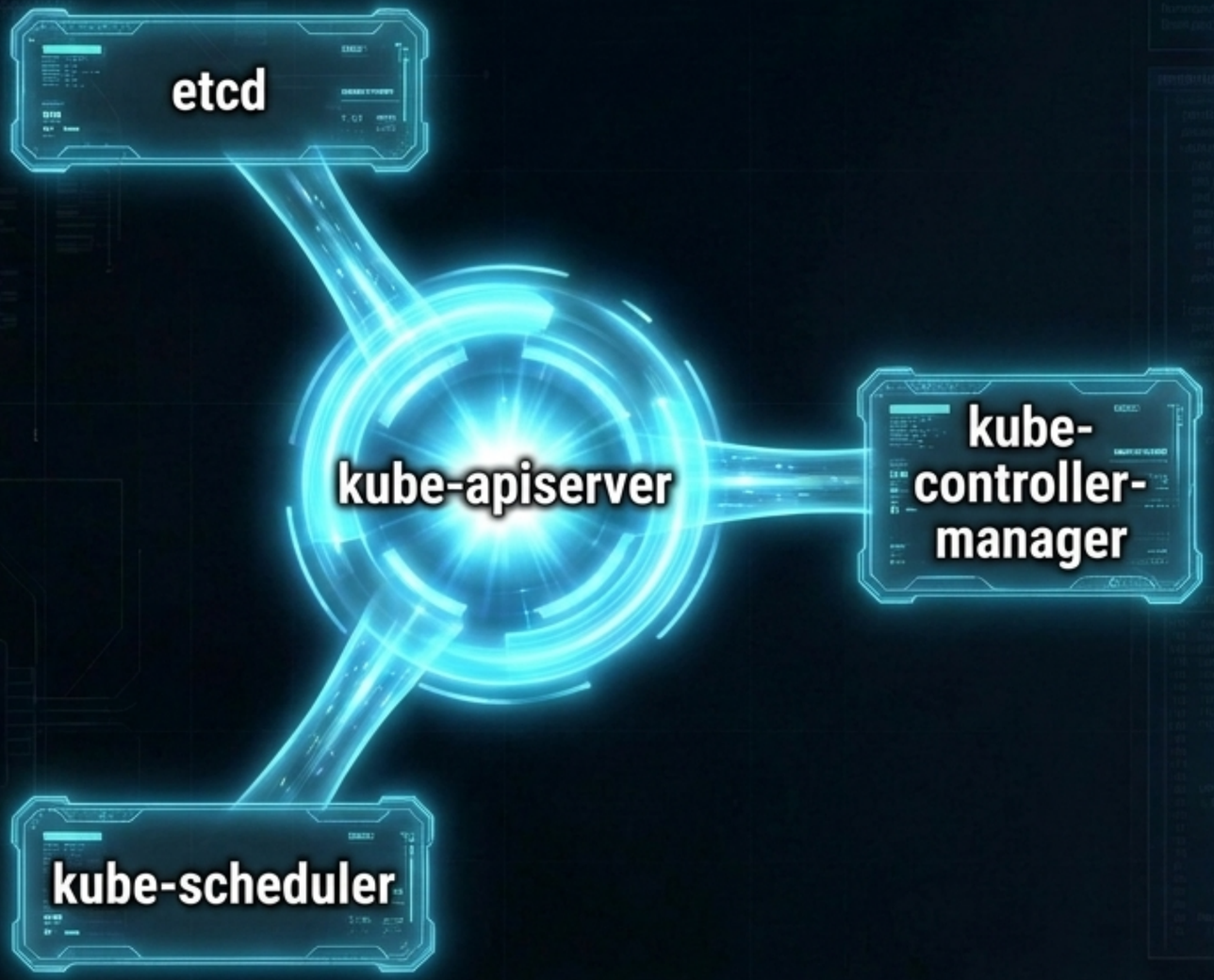


It operates as a set of independent, composable, and relentless control loops.

Level-Triggered vs Edge-Triggered: K8s does not rely on catching fleeting events (edge). It constantly compares the current state of the system against the requested state (level) and relentlessly drives reality toward alignment.

Gemini Notebook

The Brain of the Cluster



API Server: The only component permitted to speak to etcd. A pure REST interface that validates and mutates state.

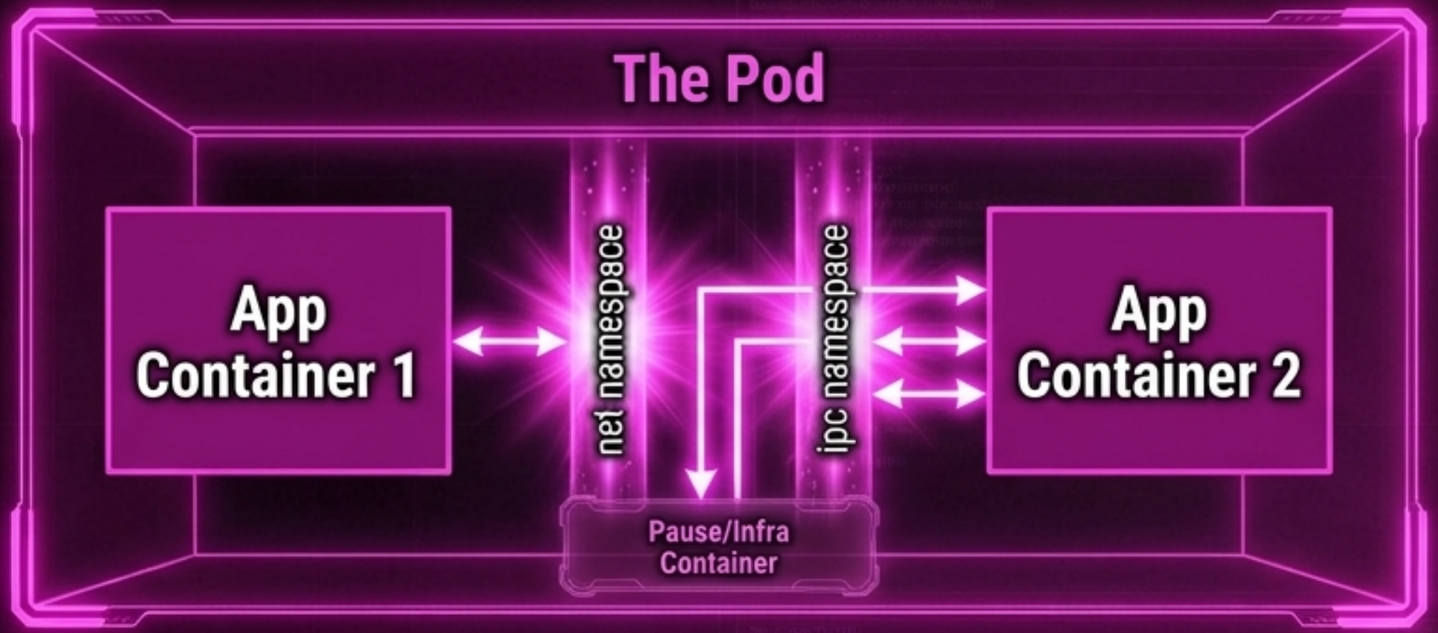
etcd: Raft-backed distributed key-value so. The definitive single source of truth for the cluster.

Scheduler: Evaluates node resources (CPU/Mem limits), taints, tolerations, and affinity rules to bin-pack pending Pods onto optimal Nodes.

Controller Manager: The engine running the continuous DaemonSet, Deployment, and ReplicaSet reconciliation loops.

Gemini Notebook

Anatomy of a Pod



- A Pod is the atomic scheduling primitive containing one or more tightly coupled containers.
- **The Pause Container**: Every Pod boots an invisible ‘pause’ container first. Its singular job is to instantiate and hold the net and ipc namespaces open.
- **Shared Identity**: This architecture allows app containers in the same Pod to crash and restart without losing the Pod’s IP address, letting them communicate seamlessly via localhost.

Gemini Notebook

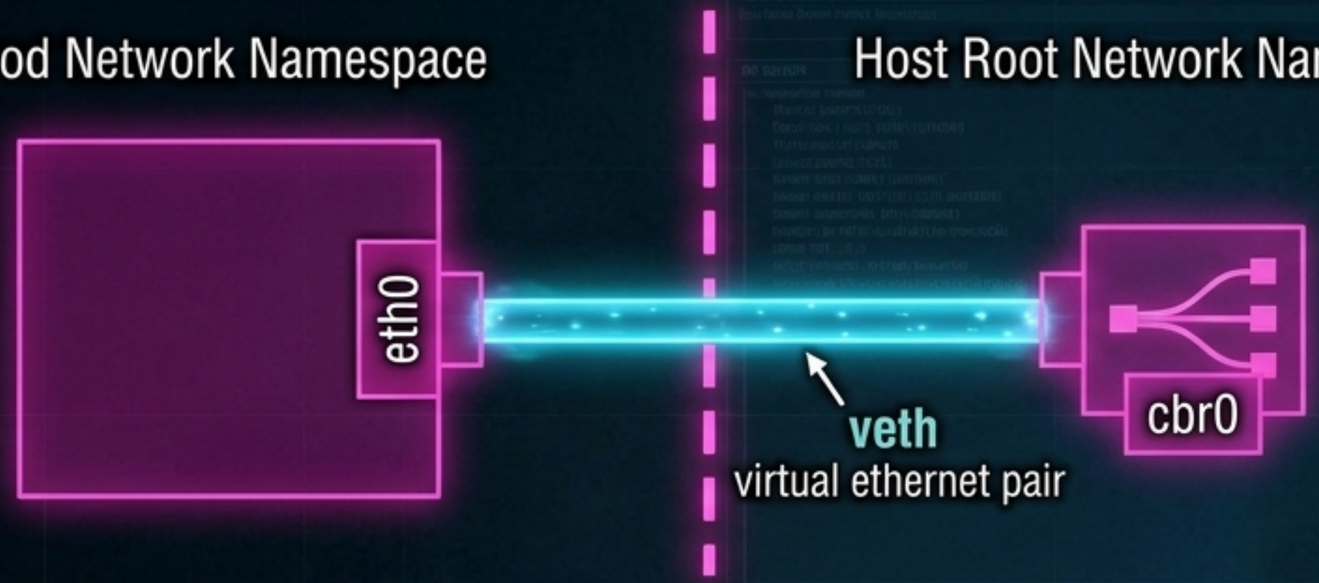
Kubernetes Networking Model



- The K8s mandate requires a globally Flat Pod IP model.
- **Rule 1**: All Pods can communicate with all other Pods without NAT.
- **Rule 2**: Agents on a node (Kubelet) can communicate with all Pods on that node.
- **Identity Integrity**: The IP address that a Pod sees itself as is the exact same IP address that all other resources see it as.

Gemini Notebook

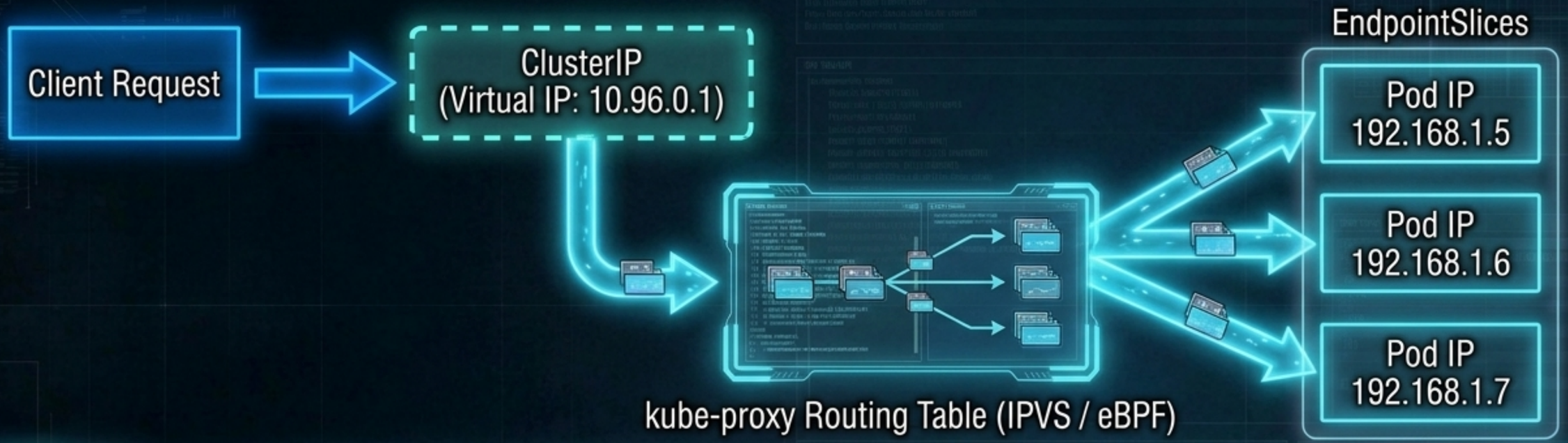
Node-Level Networking: veth & CNI



- **veth pairs**: Virtual ethernet cables. One end lives in the isolated Pod namespace, providing eth0, while the other end lives in the host's root namespace.
- **CNI Plugins**: The Container Networking Interface (e.g., VPC CNI, Calico, Cilium). Invoked by the runtime to physically wire up these veth pairs, allocate the IP, and configure routing tables and eBPF maps.

Gemini Notebook

Service Discovery & ClusterIP



- **Ephemeral Reality:** Pod IPs are highly ephemeral and constantly changing. Services provide a stable, long-lived Virtual IP (ClusterIP).
- **ClusterIPs Aren't Real:** They do not belong to any physical network interface on any machine.
- **Kernel Traps:** kube-proxy programs the node's kernel to trap traffic destined for the virtual IP and DNAT (Destination Network Address Translation) it to the actual, live Pod IPs tracked via EndpointSlices.

Gemini Notebook

Ingress & Gateway API

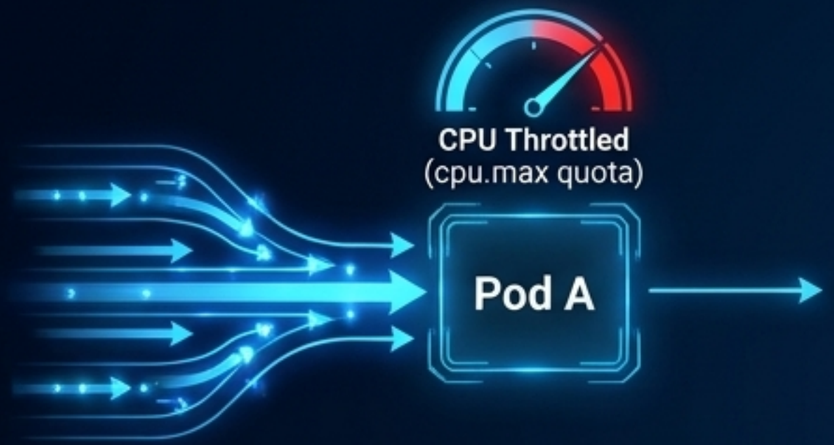


- **L4 Services:** Standard Services (type: LoadBalancer) provide basic TCP/UDP access.
- **Ingress Controllers:** Provide protocol-aware HTTP/HTTPS L7 routing based on URIs, hostnames, and specific paths.
- **Gateway API:** The modern evolution of Ingress, offering dynamic infrastructure provisioning and advanced traffic routing (canary, mirroring) natively defined in core K8s objects.

Gemini Notebook

Production Failure Mode 1: The Liveness Cascade

Step 1: CPU Throttling



Pod A receives traffic, CPU usage spikes, hits quota.

Step 2: Liveness Failure & SIGKILL



Liveness probe times out due to CPU latency, Kubelet murders the pod.

Step 3: Cluster Cascade & Darkness



Remaining pods are overwhelmed, their liveness probes fail, leading to total cluster collapse.

- Liveness probes check if an app is dead and requires restarting.
- Under heavy load, strict CPU CFS quotas cause artificial multi-threaded latency.
- If artificial latency exceeds the probe timeout, K8s murders its own struggling (but healthy) pods, actively destroying its own capacity under load.
- **The Fix:** Strictly separate Readiness (stop sending traffic) from Liveness (restart the process).

Gemini Notebook

Production Failure Mode 2: Image Pull Cold Starts



- **The Trigger:** HPA detects a CPU spike and requests new pods. The Cluster Autoscaler provisions a new Node VM.
- **The Trap:** If the OCI image is a massive monolithic tarball, downloading the SHA-256 layers and unpacking them via OverlayFS bottlenecks heavily on node disk I/O.
- **The Failure:** The cold start takes 3+ minutes; the initial traffic spike has already caused cascading timeouts before capacity arrives.
- **The Fix:** Slim base images, lazy-pulling (eStargz), or DaemonSet image pre-caching.

Gemini Notebook

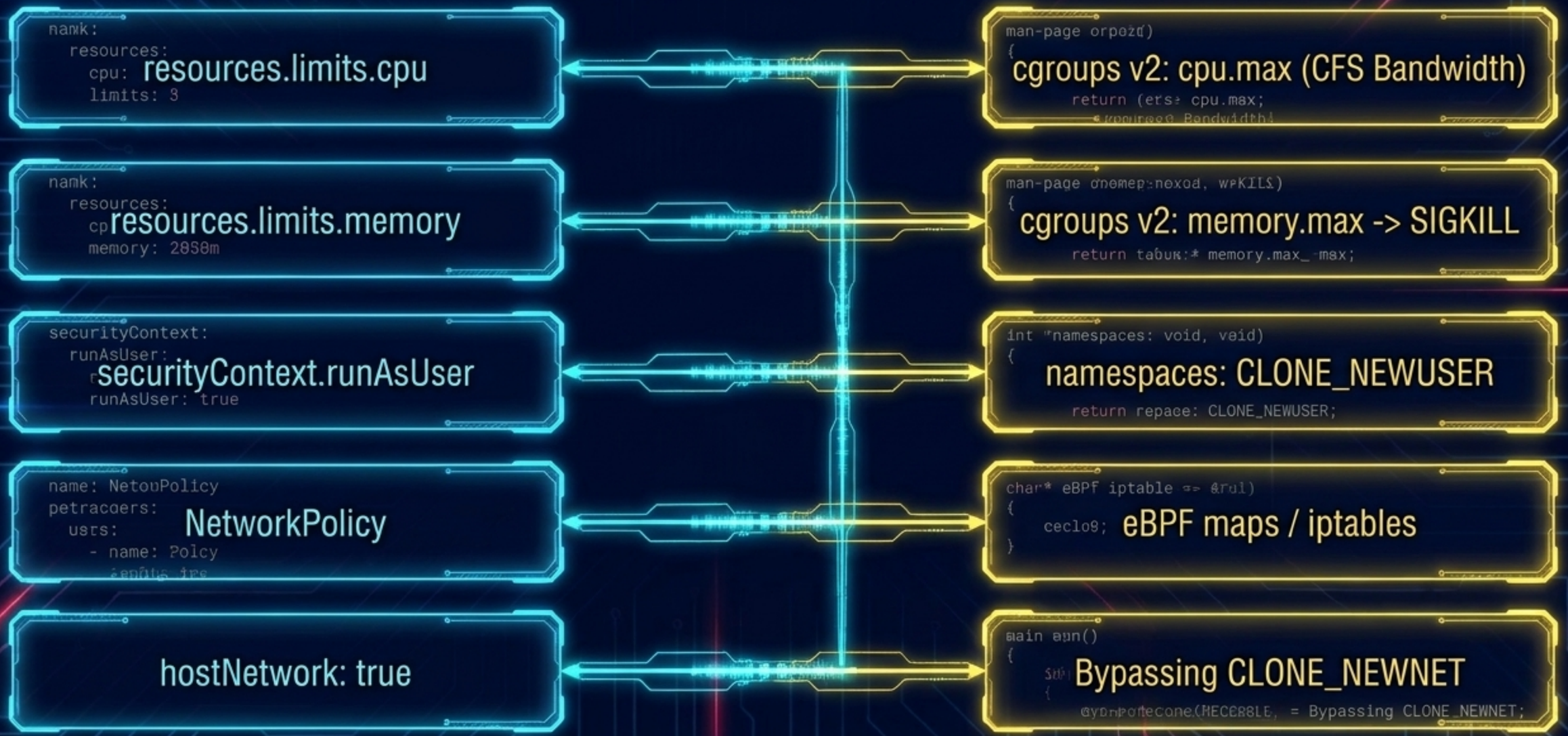
Compute Selection Matrix

	Virtual Machines (EC2)	Managed Containers (ECS / Cloud Run)	Kubernetes (EKS / GKE)
Operational Overhead	Highest	Lowest	Highest
Infrastructure Control	Highest	Medium	High
Ecosystem Extensibility	Low	Medium	Highest

- VMs: Best for static, monolithic workloads requiring strict kernel-level isolation and custom OS tuning.
- Managed Containers: Best for stateless web applications, event-driven APIs, and rapid product iterations.
- Kubernetes: Best for complex microservice architectures requiring custom controllers, specific CNI implementations, and planetary-scale orchestration.

Gemini Notebook

Synthesis: From Kubernetes to Kernel



Gemini Notebook

End of Line & References



The abstraction is a luxury. Knowing how it breaks is an engineering necessity.

Required Reading:
- man 7 namespaces, man 7 cgroups, man 7 cgroup_namespaces
- OCI Runtime and Image Specifications (opencontainers.org)
- Kubernetes Documentation: Concepts -> Architecture -> Workloads

Deck compiled for S02E05. Fully self-contained artifact.

Gemini Notebook