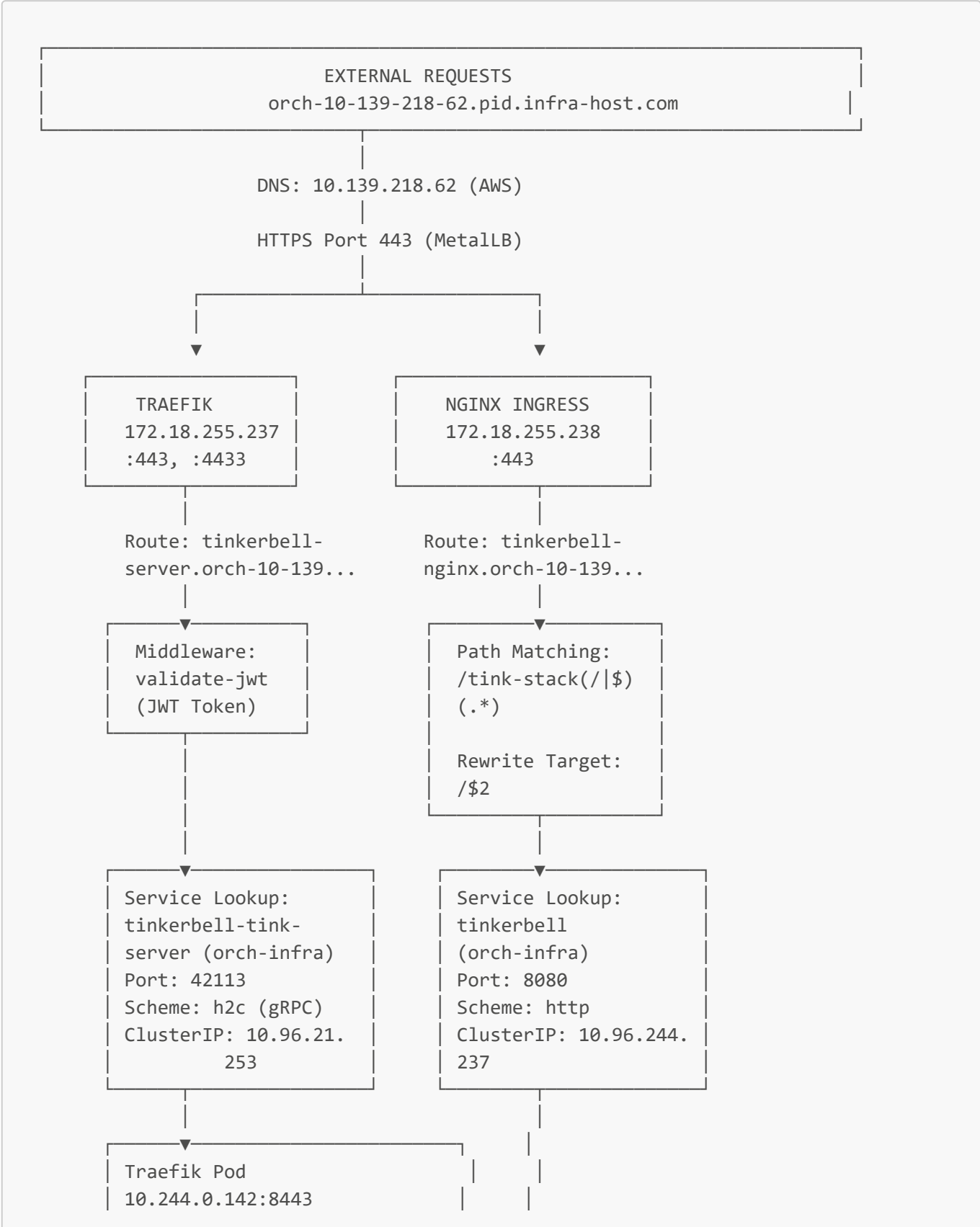
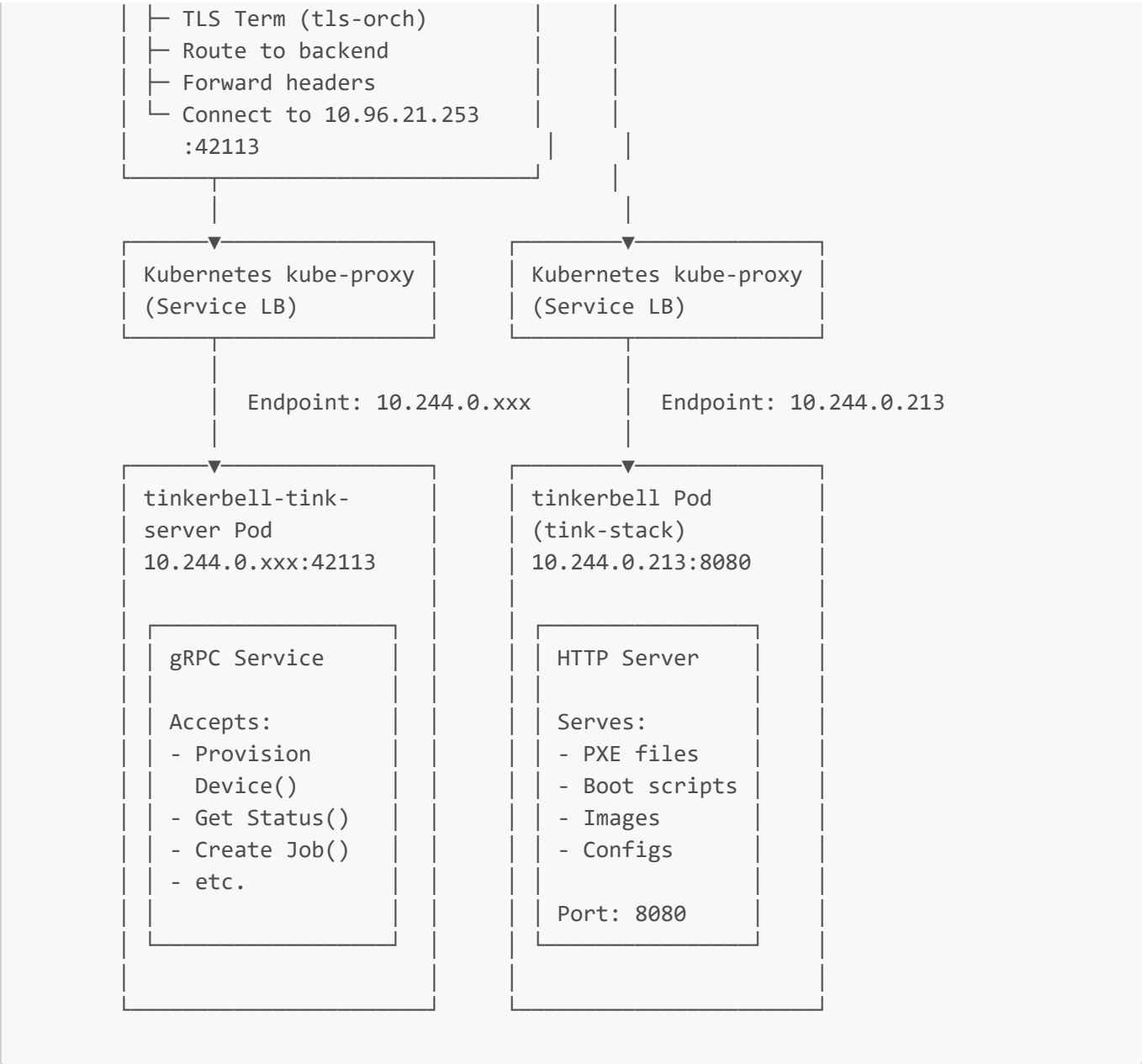


Tinkerbell Request Flow Visualizations

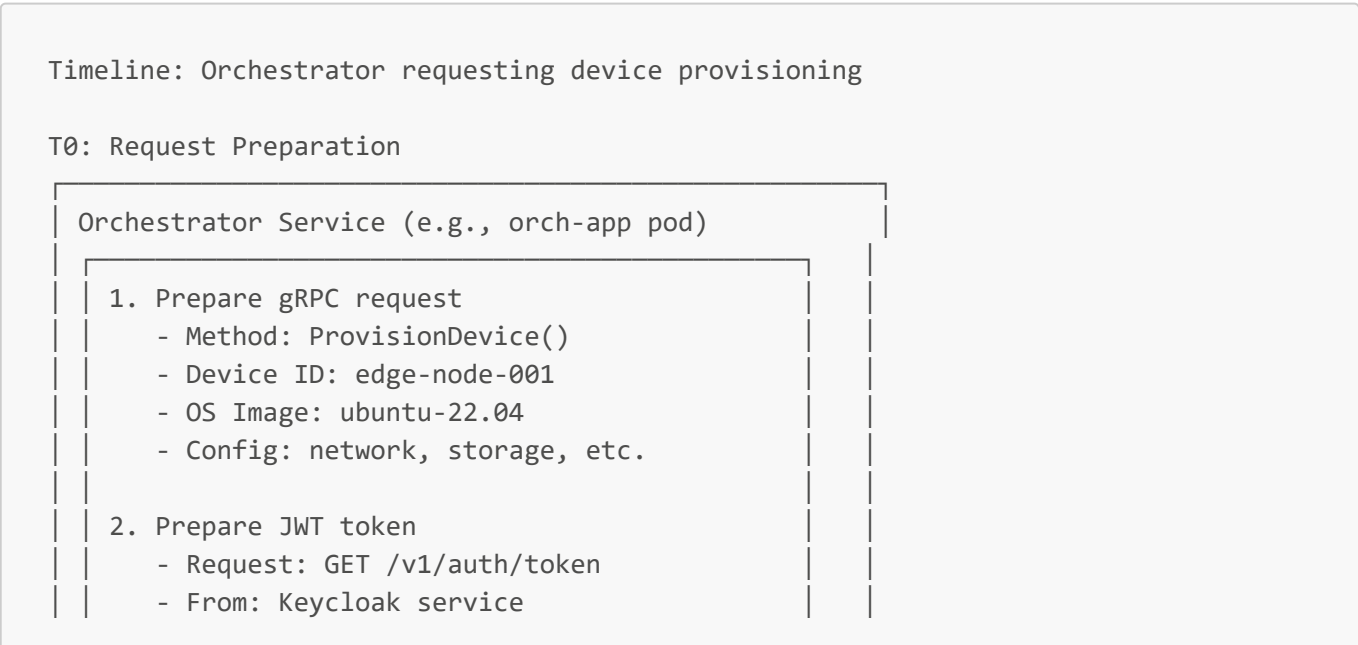
Detailed Visual Analysis of Tinkerbell Call Propagation

1. COMPLETE TINKERBELL ARCHITECTURE DIAGRAM





2. TINKERBELL REQUEST SEQUENCE (gRPC Path)



- Response: Bearer <jwt_token>
- 3. Create HTTP/2 request
 - Headers:
 - Authorization: Bearer <jwt_token>
 - Content-Type: application/grpc
 - User-Agent: grpc-go/...
- 4. Resolve DNS
 - Query: tinkerbell-server.orch-10-...
 - Response: 10.139.218.62 (AWS host)



T1: Transport Layer

Network Stack

1. TCP 3-way handshake
 - Client: SYN
 - Server (10.139.218.62:443): SYN-ACK
 - Client: ACK
2. TLS Handshake
 - Client: TLS ClientHello
 - ├ TLS Version: 1.3
 - ├ Supported Ciphers
 - ├ Extensions
 - └ Server Name Indication (SNI):
tinkerbell-server.orch-10-...
 - Server: TLS ServerHello
 - ├ Selected Cipher
 - ├ Certificate: tls-orch
 - └ CN: orch-10-139-218-62...
 - └ SAN: *.orch-10-139-218-62...
 - └ Key Exchange
 - Client: Finished
 - Server: Finished
 - Result: TLS 1.3 tunnel established ✓



T2: MetalLB Load Balancer (Host network)

- AWS Instance (10.139.218.62)
 - ├ Packet arrives: dest_port=443

MetalLB Daemon

- └ Check external addresses: 172.18.255.*
- └ Lookup: Does 172.18.255.237 exist on node?
 - └ YES (bridge interface added by MetalLB)
- └ Map to Kubernetes service:
 - └ Service: traefik (orch-gateway)
 - └ ClusterIP: 10.96.119.149
- └ Translate packet:
 - └ OLD: 172.18.255.237:443
 - └ NEW: 10.96.119.149:443
 - └ Forward to kube-proxy



T3: Kubernetes Service Layer (kube-proxy)

kube-proxy (on kind-control-plane node)

1. Intercept packet:
Destination: 10.96.119.149:443 (Traefik svc)
2. Query Endpoints:
Service: traefik (orch-gateway/traefik)
 - └ Endpoints: [10.244.0.142:8443]
 - (Only 1 replica in cluster)
3. Load Balance:
 - └ Select endpoint: 10.244.0.142
4. DNAT (Destination NAT):
OLD: 10.96.119.149:443
NEW: 10.244.0.142:8443
 - └ Forward to Traefik pod



T4: Traefik Pod Processing

Traefik Pod (orch-gateway/traefik-dccbf764d-bgw9w)
Container IP: 10.244.0.142
Port: 8443 (websecure entrypoint)

STEP 1: Receive TLS Connection

- └ TCP packet arrives on :8443
- └ TLS record received (encrypted payload)
- └ Connection established ✓

STEP 2: TLS Decryption

- | Load private key: tls-orch secret
- | Decrypt payload (ECDSA P-384 key)
- | Verify MAC
- | Plaintext HTTP/2 frame ready ✓

STEP 3: Parse HTTP/2 Request

- | Frame type: SETTINGS, HEADERS, DATA
- | Extract headers:
 - | :method = POST
 - | :path = /tinkerbell.hardware.v1.API/...
 - | :authority = tinkerbell-server.orch-...
 - | content-type = application/grpc+proto
 - | authorization = Bearer <jwt_token>
- | Extract body: protobuf encoded message

STEP 4: IngressRoute Matching

- | Iterate all 31 IngressRoutes
- | Check Host header: tinkerbell-server.orch-...
 - | MATCH: IngressRoute
name="tinkerbell-server-ingress"
- | Found matching route ✓

STEP 5: Middleware Execution

- | Middleware 1: validate-jwt
 - | Extract token: Bearer <jwt_token>
 - | Fetch JWK endpoint from middleware config
 - | Verify signature
 - | Check: exp, iss, sub, etc.
 - | ✓ Token valid
 - | Store in request context:
claims = {
 sub: "user-123",
 org_id: "org-abc",
 exp: 1706625345,
 ...
}
- | Request continues to next middleware

STEP 6: Backend Service Resolution

- | From IngressRoute spec:
 - | Service name: tinkerbell-tink-server
 - | Namespace: orch-infra

```
└─ Port: 42113
└─ Scheme: h2c (HTTP/2 Cleartext)

└─ Query Kubernetes API:
  GET /api/v1/namespaces/orch-infra/
    services/tinkerbell-tink-server
  Response:
  {
    spec: {
      clusterIP: "10.96.21.253"
    },
    status: {
      loadBalancer: {}
    }
  }

└─ Resolve to: 10.96.21.253:42113
└─ Backend ready ✓
```

STEP 7: Backend Connection

```
└─ Open new TCP connection to 10.96.21.253:42113
└─ NO TLS (h2c = HTTP/2 Cleartext)
└─ Send HTTP/2 request:
  POST /tinkerbell.hardware.v1.API/...
  Content-Type: application/grpc+proto
  grpc-encoding: gzip (if configured)
  [protobuf message body]

└─ Wait for response
└─ Connection established to backend ✓
```



T5: kube-proxy Service Resolution (Backend)

kube-proxy (backend service routing)

1. Intercept packet:
Destination: 10.96.21.253:42113
2. Query Endpoints:
Service: tinkerbell-tink-server (orch-infra)
└─ Endpoints: [10.244.0.xxx:42113]
3. Load Balance:
└─ Select endpoint: 10.244.0.xxx
4. DNAT:
OLD: 10.96.21.253:42113

NEW: 10.244.0.xxx:42113

5. Forward to pod



T6: Backend Pod Processing

```
tinkerbell-tink-server Pod
Container: tinkerbell-tink-server
Port: 42113 (gRPC)
Image: ghcr.io/tinkerbell/tink-server:v...
```

Receive gRPC request

- └─ Unpack protobuf message
- └─ Method: ProvisionDevice()
- └─ Parameters:
 - └─ device_id: "edge-node-001"
 - └─ os_image: "ubuntu-22.04"
 - └─ config: {...}
- └─ (Optionally) extract claims from headers
 - └─ Claims would be added by middleware

Process request

- └─ Validate device ID exists
- └─ Check authorization:
 - └─ Can user provision this device?
- └─ Create provisioning job:
 - └─ INSERT INTO jobs Table
 - {
 - id: "job-uuid-123",
 - device_id: "edge-node-001",
 - user_id: "user-123",
 - status: "PENDING",
 - created_at: now(),
 - config: {...}
 - }
- └─ Communicate with tink-controller:
 - └─ Send: WorkflowStart event
 - {
 - job_id: "job-uuid-123",
 - action: "provision",
 - device: {...}
 - }
- └─ Return gRPC response:

```
{
  job_id: "job-uuid-123",
  status: "ACCEPTED",
  message: "Provisioning initiated"
}
```

Send response back through chain

- └─ Serialize protobuf message
- └─ HTTP/2 response frames
- └─ Status: gRPC Status OK (0)
- └─ Send to Traefik pod



T7: Response Path Back to Client

Reverse path through Traefik

1. Traefik receives response from backend
2. Forwards HTTP/2 frames back to client
3. Re-encrypts with TLS (tls-orch certificate)
4. Sends encrypted response through MetalLB
5. Response reaches client:

```
{
  job_id: "job-uuid-123",
  status: "ACCEPTED"
}
```

✓ Provisioning initiated!

3. HTTP FILE SERVING PATH (Tinkerbelle-Nginx)

PXE Boot Client (Bare Metal Machine)

- └─ Request: GET /tink-stack/hook.ipxe
- └─ URL: https://tinkerbelle-nginx.orch-10-139-218-62.pid.infra-host.com/tink-stack/hook.ipxe
- └─ DNS: tinkerbelle-nginx.orch-10-139-218-62... → 10.139.218.62
- └─ TLS Connection to 10.139.218.62:443 (MetalLB)

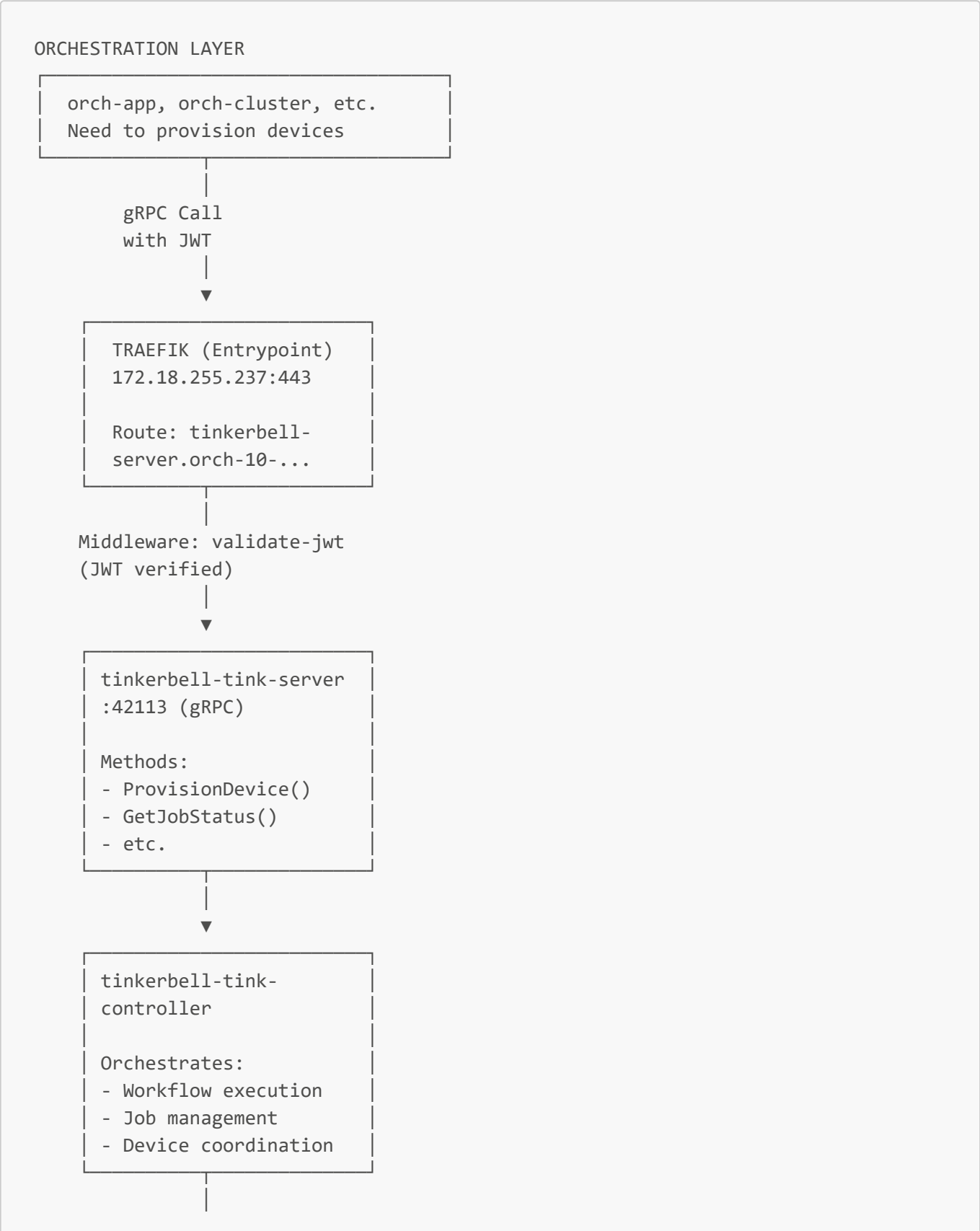
```
|
|─ MetalLB:
|   └─ Map 172.18.255.238 → nginx-controller (orch-boots)
|
|─ Nginx Controller Pod (10.244.0.144:443)
|   |
|   └─ TLS Termination (nginx default cert)
|       └─ Decrypt request
|
|   └─ Host Matching:
|       └─ Host = tinkerbell-nginx.orch-10-139-218-62... ✓
|
|   └─ Path Matching & Rewriting:
|       └─ Original: /tink-stack/hook.ipxe
|       └─ Regex: /tink-stack(/|$)(.*)
|           └─ Capture: $1='/', $2='hook.ipxe'
|
|       └─ Rewrite Target: /$2
|           └─ Result: /hook.ipxe
|
|       └─ ✓ Match!
|
|   └─ Backend Service Lookup:
|       └─ Service: tinkerbell
|       └─ Port: 8080 (HTTP)
|       └─ ClusterIP: 10.96.244.237
|       └─ Endpoint: 10.244.0.213:8080
|
|   └─ Nginx connects to 10.96.244.237:8080
|       └─ kube-proxy LB to: 10.244.0.213:8080
|
|       └─ Send HTTP request:
|           GET /hook.ipxe
|           Host: tinkerbell
|           User-Agent: curl/...
|
|─ tinkerbell Pod (10.244.0.213:8080)
|   |
|   └─ HTTP Server receives request
|   └─ Lookup file: /hook.ipxe
|   └─ File exists: /var/tink/ipxe/hook.ipxe
|
|   └─ Read file content
|   └─ Send response:
|       HTTP/1.1 200 OK
|       Content-Type: application/octet-stream
|       Content-Length: 8192
|       [Binary file content]
|
|   └─ Connection close
|       |
|       └─ Nginx receives file
|       └─ Re-encrypt with TLS
|       └─ Send to PXE client
```

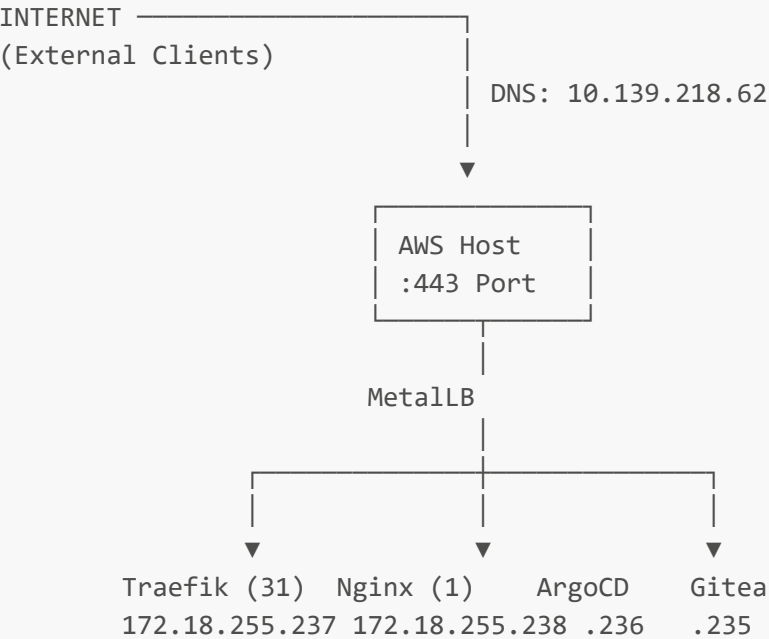
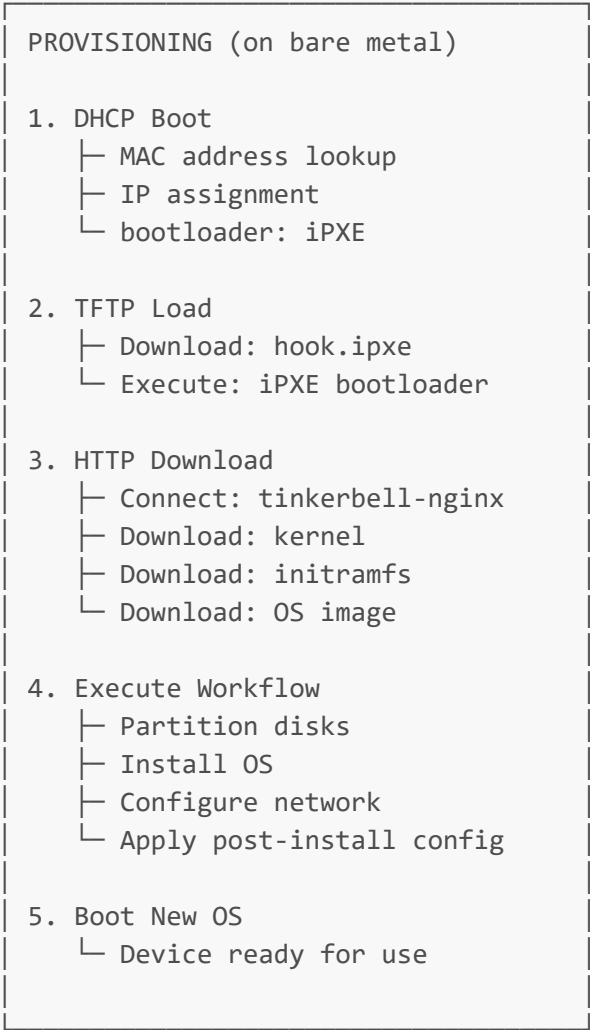
- |
 - └ PXE Client receives:
 - └ HTTPS 200 OK
 - └ File content (hook.ipxe)
 - └ Write to memory
 - └ Execute iPXE bootloader
 - |
 - └ Load next stage
 - └ Download kernel
 - └ Download initramfs
 - └ Begin provisioning...

4. DUAL-PATH FLOW MATRIX

TINKERBELL DUAL ROUTING PATHS
<div><div>PATH 1: Traefik → gRPC (42113, h2c)</div><div><div>Use Case: Service-to-Service Provisioning</div><div><ul style="list-style-type: none">└ Orchestrator calls: tinkerbell-server.orch-10-...└ Protocol: gRPC (Protocol Buffers)└ TLS: Yes (HTTPS → h2c conversion)└ Auth: JWT required└ Rate Limit: None└ Backend Service: tinkerbell-tink-server (42113)└ Use Cases:<ul style="list-style-type: none">└ Create provisioning jobs└ Query job status└ Apply configurations└ Manage workflows</div></div></div>
<div><div>PATH 2: Nginx → HTTP (8080)</div><div><div>Use Case: File Serving for PXE Boot</div><div><ul style="list-style-type: none">└ Boot Client calls: tinkerbell-nginx.orch-10-...└ Protocol: HTTP/REST└ TLS: HTTPS only (port 443 → 8080)└ Auth: None (public)└ Rate Limit: 500 req/s, 70 connections└ Backend Service: tinkerbell (8080)└ Use Cases:<ul style="list-style-type: none">└ Serve PXE bootloader (iPXE)└ Serve kernel & initramfs└ Serve hook scripts└ Serve OS images└ Serve device configs</div></div></div>

5. SERVICE COMMUNICATION MAP





6. PROTOCOL STACK COMPARISON

