# Chapter 1: Getting Started

# Section 1: Requirements

## Recommended VM Specifications

Since we're not simulating the scale of a full-blown ISP, we can afford to keep the VMs relatively light:

1. **VyOS VM**:
   - **CPU**: 2 cores
   - **RAM**: 1 GB (2 GB recommended)
   - **Disk**: 8 GB
   - **Network**: 1 NIC connected to the real-world bridge, 1 NIC to the core network bridge, and additional NICs for each virtual ISP bridge.
2. **DNS Root VM**:
   - **CPU**: 1 core
   - **RAM**: 512 MB (1 GB recommended)
   - **Disk**: 4 GB
   - **Network**: 1 NIC connected to the core network bridge.
3. **Stratum 1 Time Server VM**:
   - **CPU**: 1 core
   - **RAM**: 512 MB (1 GB recommended)
   - **Disk**: 4 GB
   - **Network**: 1 NIC connected to the core network bridge.
   - *Hint*: There's something unique planned for this VM that will add a touch of realism to the time synchronization setup. Stay tuned.

## Knowledge Prerequisites

To get the most out of this project, you should be comfortable with:

- **Basic Networking**: IP addressing, routing concepts, and network segmentation.
- **Linux Command Line**: Many configurations will require command-line interaction, especially when setting up services on VMs.
- **Virtualization**: Familiarity with creating and managing VMs within Proxmox.

## Time and Resource Estimates

1. **Time Commitment**:

- **Initial Setup**: Approximately 1-2 hours for configuring Proxmox, setting up bridges, and creating initial VMs.
- **Configuration and Testing**: Expect to spend 3-4 hours on setting up core services and ensuring everything is communicating correctly.
- **Full Deployment**: Over the course of the weekend, you should be able to bring the entire virtual internet ecosystem to life.

2. **Resource Management Tips**:
   - **Use Snapshots**: Take snapshots of your VMs at key configuration points. This allows you to roll back quickly if something goes wrong.
   - **Monitor Resource Usage**: Keep an eye on CPU, RAM, and disk usage through Proxmox's interface. Adjust allocations as needed.

# Conclusion

With these requirements in hand, you're ready to start building. This section ensures you have a clear understanding of what's needed and how each piece will fit together as we proceed. The next step is to lay out the network architecture, mapping out how our virtual ISPs will interact and how the core services will keep everything running smoothly. Let's get into the design phase!

# Section 2: High-Level Network Architecture and Design

Before we start configuring VMs and services, it’s essential to understand the overall architecture of the virtualized internet ecosystem we are building. This section provides a high-level view of how everything will be designed, including the key components, network topology, and security considerations. Think of it as the blueprint that will guide us through the technical setup in the following chapters.

## 1. Network Topology Overview

Our virtual internet ecosystem will mimic the real-world structure of an ISP network, but on a smaller, more manageable scale. At its core, the system will consist of:

- **Core Services**: These include our primary router, DNS root server, and Stratum 1 time server.
- **Network Bridges**: The connections between internal VMs, virtual ISPs, and the real world. Each service and ISP will have a dedicated segment, ensuring clean, isolated networking.
- **Virtual ISPs**: Independent network segments, each functioning as its own isolated ISP environment with its own routing, DHCP pools, and configurations.

Here’s a breakdown of how the network will be organized:

- **Core Network (vmbr-core)**: The heart of our ecosystem, connecting all core services.
- **Real-World Bridge (vmbr0)**: Provides connectivity between our virtual environment and the external internet, allowing certain services to be accessible publicly.
- **ISP-Specific Bridges (vmbr-isp1, vmbr-isp2, etc.)**: Each virtual ISP will have its own bridge, creating isolated segments that can be routed and managed independently.

## 2. Key Components and Their Roles

1. **Core Router (VyOS)**
   - The VyOS router will serve as the backbone of the network. It will handle:

    - **Dynamic Routing (BGP)**: Enabling communication between different segments of the network and simulating real-world routing behavior.
    - **Network Segmentation**: Isolating traffic between virtual ISPs, core services, and external networks.
    - **Security Policies**: Acting as a firewall and enforcing rules on how data moves between segments.
2. **DNS Root Server**
    - This VM will manage domain name resolution within the virtual ecosystem.
    - **Internal DNS Management**: Each virtual ISP will rely on this DNS root for resolving domain names, ensuring smooth communication within the network.
    - **Scalability**: Configured to handle both internal services and external DNS requests, simulating real-world DNS infrastructure.
3. **Stratum 1 Time Server**
    - A Stratum 1 server is a precise time source, usually directly connected to a GPS or atomic clock. In our setup, this VM will:
        - **Provide Accurate Time Synchronization**: Ensuring all services and VMs are in sync, which is crucial for network operations, security protocols, and troubleshooting.
        - **Hint**: There's a special feature we'll implement to give this server a realistic touch, making our virtual ecosystem's time synchronization as precise as possible.
4. **Root Certificate Authority**

- The Root Certificate Authority (CA) will serve as the central entity for issuing and managing digital certificates within our virtual ecosystem. It ensures that all internal services can communicate securely by providing trusted certificates for encryption protocols like HTTPS and TLS.
- **Purpose**: Setting up our own Root CA allows us to control the security infrastructure, enabling encrypted connections across the network. This is essential for protecting data, establishing trust between services, and mimicking real-world ISP security practices.

1. **Virtual ISPs**
    - Each virtual ISP will be an isolated environment, complete with its own:
        - **Routing and DHCP**: Independent configurations, simulating how different ISPs operate separately yet can be managed centrally.
        - **Custom Services**: DNS and DHCP servers for each ISP will be configured to allow for unique setups, providing a more realistic ISP simulation.

# 3. Bridges and Virtual Network Segmentation

1. **Core Network Bridge (vmbr-core)**

- Central to internal communication between core services, such as the VyOS router, DNS root server, and time server.
- Acts as the backbone, ensuring all core components are tightly integrated and efficiently communicating.

2. **Real-World Bridge (vmbr0)**
   - This bridge connects the virtual environment to the external internet.
   - **External Access**: Necessary for functions like downloading software updates, public-facing monitoring, and allowing selected external services to interface with the internal network.
   - **Secure Routing**: Traffic between the real-world network and internal systems will be strictly managed via VyOS and OPNsense.
3. **ISP-Specific Bridges (vmbr-isp1, vmbr-isp2, etc.)**
   - Each virtual ISP gets its own bridge, creating isolated network segments.
   - **Isolation and Security**: This ensures that traffic is contained within its segment unless explicitly routed through the core network.
   - **Scalability**: Easy to add new ISP segments by creating additional bridges, enabling future expansion without disrupting the existing setup.

# 4. Security Considerations and Traffic Flow

1. **Traffic Isolation and Management**
   - **NAT and Firewalls**: VyOS will manage Network Address Translation (NAT) and firewall rules, ensuring that internal segments are secure and external access is controlled.
   - **Traffic Filtering**: Rules will be configured to allow or deny traffic between virtual ISPs, core services, and the external world based on defined security policies.
2. **OPNsense and HAProxy Integration**
   - **OPNsense**: Will function as a secondary layer of security, handling more advanced firewall and filtering tasks. It will also provide VPN services and further segment internal traffic.
   - **HAProxy**: Responsible for load balancing and managing traffic to and from the public-facing services hosted within the ecosystem, ensuring smooth, reliable access for monitoring tools like Grafana and Zabbix.

# 5. Visualization and Monitoring

1. **Internal and External Monitoring**
   - Tools like **Zabbix**, **Grafana**, and **OpenSearch** will be used to track key metrics across the network, including traffic flows, resource usage, and performance indicators.

- **Live Dashboards**: Accessible both internally and (selectively) from the external internet, giving a real-time view of the network's health and activity.

2. **Secure Public Access via NGINX Proxy**
   - **Traffic Routing**: Using **NGINX** at a remote server (e.g., Linode) to securely route external requests to the internal monitoring systems.
   - **Data Security**: Ensures that only specific data and metrics are exposed, while maintaining a strong security posture across the network.

# Conclusion

This architecture lays out the groundwork for everything we're about to build. By understanding the high-level design and purpose of each component, you'll have a clearer path forward when it's time to configure the VMs, set up routing, and deploy services. The network topology and segmentation we've planned ensure scalability, security, and realistic simulation of real-world ISP environments. Now that we've covered what's needed and how it will all fit together, it's time to start building in Chapter 2.

We'll begin by setting up the core services, starting with the VyOS router, DNS root server, and our special Stratum 1 time server. This foundation will pave the way for the rest of the project, so let's get ready to dive in.