Introduction

I’ve noticed a lot of people struggling with RouterOS setups, so I made this lab. These configurations can be used for learning or as a base for your own networks. MikroTik’s help page is solid overall, but it can be tough for newcomers to grasp the concepts and follow isolated examples, so a full network export might be easier to understand. The configurations cover basic ideas that are essential (or useful) for SOHO and larger networks. I’d recommend checking out these concepts on MikroTik’s help page, searching online, or asking a chatbot if you’re unsure - then use these configurations as a reference while experimenting or setting up your own networks.

For this lab I used devices I had at home, but you can use almost any MikroTik hardware. RouterOS is versatile, though its capabilities are often limited by the device’s hardware and feature set. For example, you can apply the R1 configuration to even cheaper devices like the hEX lite or the most powerful CCR so far - the CCR2116. As wireless devices, you can use any of the Wi-Fi 6 (802.11ax) models, whether indoor or outdoor, or IPQ40xx-based Wi-Fi 5 (802.11ac) devices. Other devices will be incompatible with Wave2 CAPsMAN, but they can still be used as standalone devices or with legacy CAPsMAN. I recommend getting familiar with MikroTik devices, license level limitations, and their hardware characteristics to understand their intended use cases. For instance, AP (Access Point) and Point-to-Point (PTP) link configurations are similar, but on a PTP device, License Level 3 restricts you to just one wireless connection. This means such devices can’t be used as APs for Point-to-Multipoint (PTMP) links - only for PTP links. Additionally, PTP devices usually have more powerful and directional signal beams, which aren’t ideal for PTMP setups.

Configurations are available in two versions: one with VLAN support and one without. The non-VLAN version is made for simple networks that require a single broadcast domain, or for users just starting out. The VLAN version divides a single physical network into multiple logical segments, creating multiple broadcast domains. This is useful for isolating IoT devices or setting up a guest network at home. In the VLAN version, we’ll configure a guest network. Make sure you read the detailed comments below.

Included concepts

In the following configurations, I've included the following concepts:

- Topology plan. Every network needs a topology plan to ensure smooth packet flow and scalability. Avoid undocumented "bird's nests" with multiple routers, these can mess up your local network. Sketch out a clear topology, label all ports, and stick to it. Unused ports on network devices can be repurposed for local access or future expansion, as long as they're properly configured.
- Public internet access is provided via LTE. LTE1 is configured in pass-through mode, with a TTL mangle rule applied on R1. This configuration can be reverted to use the ether1 port on R1 as a regular WAN port by making a few changes to the settings (as described below).
- STP (Spanning Tree Protocol) configuration is important. As MikroTik's help page says, not quite: "You need to configure STP if your network has more than two bridges." So, if you have more than two network devices on your network, or if you're connecting endpoints via STP-aware bridges (like virtual machine hosts or Docker setups), then it's mandatory. Otherwise, you might run into ports randomly locking up. VLAN configurations use MSTP (Multiple Spanning Tree Protocol), which is VLAN-aware, while RSTP (Rapid Spanning Tree Protocol), used in non-VLAN setups, is the default mode and isn't VLAN-aware.
- VLANs are divided into two groups: the main network (VLAN 10) and the guest network (VLAN 20). Additional VLANs can be configured as needed, like an IoT network alongside the main and guest networks. Inter-VLAN communication is isolated, and the guest network has a 10 Mbps limit in both directions. Note: Some older or specific devices use a different VLAN implementation type that isn't covered here, make sure to check device compatibility before applying these settings.
- Simple IP addressing provided by DHCP server(s), including static leases. All network devices have a static IP address. For example, an (imaginary) web server is assigned a static lease, with port forwarding configured to port 80 (HTTP). DHCP pools start at x.x.x.21, since the first addresses are reserved for static allocation.

- New Wave2 CAPsMAN is used for Wi-Fi management. It supports all current ax APs and ac lineup APs with the wifi-qcom-ac package. (Check the device support list on the help page.) However, older devices can’t be added to the new Wave2 CAPsMAN. Legacy devices still need to be managed using the older CAPsMAN version. This means you now have to run two separate management platforms, one for legacy devices and one for modern ones.
- The PTP link is configured to establish a wireless bridge on the 5 GHz band while utilizing the 2.4 GHz band as the CAPsMAN interface. The PTP link connects two APs and acts as a bridge between two locations, extending the network, like connecting a shed to a house. Ethernet ports on WBR1 can then be used for further network distribution.
- Two VPN configurations are included: Road Warrior and Site-to-Site. The Road Warrior setup is used when you need to connect to your network while not at home, such as accessing your Home Assistant instance from a phone while waiting at a bus stop or connecting to a local server from a laptop while drinking coffee at a coffee shop. It can also function as a private VPN service, eliminating the need to subscribe to a paid service — instead, you can use your own infrastructure. The Site-to-Site setup connects two remote locations, such as your home and your parents’ house. This allows you to share resources from your local network, like a media server, NAS (Network Attached Storage), or a pi-hole instance. WireGuard is used as the VPN protocol.
- Basic firewall. The default configuration firewall is fine, but I think it includes several unnecessary rules for a basic setup. For example, the default rules "defconf: accept to local loopback (for CAPsMAN)" and "defconf: drop all from WAN not DSTNATed" aren’t actually needed in my opinion. Plus, I added an input rule to allow VPN connections, etc.
- Interface lists on R1 are used for CAPsMAN ACLs and firewall filters, including NAT and LTE mangle rules. Also, R1 and LTE1 are configured to restrict device discovery and MAC-based access to the LAN list, which helps prevent local network information from leaking to the public network. VLAN version use these lists more extensively.
- RoMON is configured for convenient access. For example, when connecting to a local network over a VPN, you can log in to the server and access all network devices - even if they don’t have an IP address assigned or have

been lost, but are still reachable at the MAC level.

- Other various things like time zone, device naming, Wi-Fi debug logs on R1 for improved troubleshooting, fixed L3 MTU values on bridges, etc.

Comments and tips

Detailed comments and tips about these configurations:

- STP – The core switch (SW1) is configured as the root bridge, so all other bridges are assigned priority values that point to it. By default, all STP-aware bridges have a priority of 8000 (32768), lower numbers mean higher priority. If a new bridge is added to this network without configuration, it won't disrupt the current setup, its default priority will be lower than any in the network. Just make sure to set proper priorities when expanding the network. Also, remember that port priority, path cost, MAC addresses, and a few other factors are involved in the election process.

  Edge status on bridge ports is set manually. This ensures BPDUs (Bridge Protocol Data Units) are either received or blocked as intended. All connections between network devices are set to "edge=no", which allows BPDU packets to pass through, these are called trunk ports. Access ports, on the other hand, are set to "edge=yes", which blocks BPDUs and prevents unwanted changes to the STP topology.

  DHCP snooping and IGMP snooping are implemented on the core switch (SW1). DHCP snooping allows only designated ports to forward DHCP messages, preventing rogue DHCP servers from compromising the network. IGMP snooping ensures multicast traffic is forwarded only to ports that need it, rather than being broadcast to all. Use these features with caution, some devices don't support hardware offloading, and they can cause issues in certain setups. If you do use them, make sure they're properly configured. The larger the network, the more critical these features become. Rogue DHCP servers can appear by accident, and multicast traffic can consume a significant portion of total bandwidth in some cases.

- Here's how to disable LTE pass-through on R1 in the non-VLAN version:

  /interface/vlan/remove vlan3_passt

  /interface/vlan/remove vlan2_man

  /interface/list/member/remove numbers=0

  /interface/list/member/add interface=ether1 list=WAN

  /ip/dhcp-client/set interface=ether1 numbers=0

/ip/firewall/mangle/remove numbers=3

- Configure the LTE APN profile as needed and ensure that the modem firmware is updated. For 5G devices, the configuration should be similar or identical to LTE, though I currently don't have a 5G test device available, nor an eSIM.
- Bridges have fixed MAC addresses, they're prone to changing for various reasons, which can cause problems. L3 MTU is fixed at 1500 bytes (the most common default) because it can also be altered unexpectedly.
- 192.168.200.0/24 is used as the local network IP range in the non-VLAN setup. For VLAN configuration, 192.168.210.0/24 and 192.168.220.0/24 are assigned to VLAN 10 (main network) and VLAN 20 (guest network), respectively.
- CAPsMAN – I configure my setups using fixed interfaces instead of dynamic ones, because I prefer granular control over individual interfaces. For example, I usually set fixed channels for radios, adjust TX power as needed, change BSSID MAC addresses, specify which interface list to use, etc. This approach gives me flexibility and ensures consistent configuration.

  I've included both automatic and fixed channel presets for both bands. As a guideline, set a 20 MHz channel width for the 2.4 GHz band, and choose a width for the 5 GHz band based on your actual needs and what the environment allows. I don't recommend using maximum channel width unless necessary, narrower channels usually have less noise and can actually give you a bit more range. Pick channels that are free (no other APs nearby), or have the least neighboring networks and low channel utilization (load). Use the "Freq. Usage" tool to gather data and make an informed decision, or just go with one of the automatic channel selection profiles. Note: Not all possible fixed channels are included, only those available in Latvia.

  On security, I've set up a few profiles: WPA2, WPA2/WPA3, and WPA3. WPA2/WPA3 is the most compatible option and should be your default. When stations connect, they'll negotiate the highest supported authentication type. Use WPA2 for legacy devices or when troubleshooting Wi-Fi issues, it's older, simpler, and less secure, but has broader compatibility. It's especially useful for IoT networks, since many IoT devices don't support WPA3, and compatibility issues can arise when WPA3 and/or MFP (Management Frame Protection) are advertised in beacon frames.

WPA3 is the modern standard and should be used when your devices support it, it offers stronger encryption and better security features. Also keep in mind that Wi-Fi de-authentication attacks are easy to carry out, so make sure your PTP links are properly secured too.

The ACL example entry blocks unwanted MAC addresses from joining the network. Other examples include setting a minimum RX level per band, each band is handled separately. If a station drops below that signal threshold, it gets disconnected. This is typically used to enforce roaming at a specific signal level and to prevent "sticky clients." Use these settings carefully - only when needed. Remember: roaming is always up to the station. The decision is usually made by the client itself, based on factors like signal strength, band, channel utilization, and other info passed from the AP via management frames.

AP provisioning happens automatically based on the radio type, with interfaces created as static. The naming scheme is “AP_Identity-Band”—like AP1-2G and AP1-5G. The CAPsMAN server doesn’t use certificates. The update policy is set to "suggest-same-version". You can upload required packages to R1’s root directory, and it’ll automatically push updates to the APs.

- AP1 and AP2 have identical configurations in the non-VLAN setup, with the only difference being the number of ports in the bridge. VLAN configurations differ, as the wifi-qcom-ac package has certain limitations. Keep this in mind when setting up. The CAPsMAN server is manually defined, as L2 connections (default discovery behavior) can be unstable in some cases.

- The PTP link has a netwatch script on the WBR1 side, which I implemented to address reconnection issues when the connection to the AP drops (MikroTik confirmed this issue — a fix should land at some point). The script triggers an interface toggle, and the connection should be restored within 30 seconds.

- VPN server requires a static IP address. WireGuard configuration example is included (phone0.conf in the wg folder), which can be imported to your phone, laptop, or any other compatible device - or you can manually enter the values. This can serve as a base configuration for Road Warrior setups. Just remember to update the endpoint IP address and port (if changed) before using it.

WireGuard’s basic principle is key exchange - both sides exchange public keys, while private keys stay private. A minimal configuration file includes interface creation and peer setup.

The interface is a virtual network interface created on the device with specified parameters, including the private key. You can generate the private key when creating a new peer in RouterOS. Here’s a CLI command example for creating a new peer:

```
/interface/wireguard/peers
add allowed-address=192.168.80.3/32 comment=Laptop interface=wg1
name=peer3 persistent-keepalive=25s private-key=auto responder=yes
```

Update the comment, peer name (use a unique one or continue numbering), and set the allowed IP address - pick one after the last used (e.g., 192.168.80.2/32 was the phone’s peer IP). Note that all peers are managed under a single registry, regardless of how many WireGuard interfaces exist on the device. Once created, copy the generated private key into the interface section of your new config file. R1’s DNS server is used by default, but you can change it if needed. The MTU is set to WireGuard’s default.

The peer represents the remote side’s configuration and usually stays unchanged after initial setup. It defines the endpoint’s IP and port, keepalive interval (heartbeat ping), public key, and allowed IPs. In the config file, the default route (0.0.0.0/0) is used, meaning all traffic from the phone or laptop goes through the VPN. However, you can restrict routing to a specific range, like your local network (192.168.200.0/24) to limit the tunnel to only that range. For example, if you want to use your LTE/5G connection for general internet but only route traffic to local resources like servers or a NAS via the VPN.

Site-to-Site follows the same principles: key exchange, interface setup, and peer configuration. In this example, the remote site is the server, and the lab is the client connecting to it. The file “WG_remote_server.rsc” in the wg folder contains the necessary config for another RouterOS device. In my case, the server is my home network’s main router (WG1), so I’ve removed my public key and IP address from the config files. On RouterOS, I recommend configuring WireGuard manually - it gives you better control. As you can see on R1, there’s another WireGuard interface set up for this tunnel. Each interface serves a distinct purpose: “wg1” is used for Road

Warrior setups with its own IP range (192.168.80.0/24), traffic flow, and specific config. “wg2” is for Site-to-Site tunneling, using a separate IP range (192.168.90.0/24) and dedicated settings. WireGuard is simple yet powerful - it integrates seamlessly into RouterOS and can be used for various purposes. Yes, you can link multiple sites together, and even extend broadcast domains using VXLAN or EoIP over these tunnels - but that’s for another time.

Site-to-Site setup is similar to Road Warrior: establish a WireGuard tunnel, assign an IP address to the tunnel interface, and add a route to the other site’s IP range (192.168.200.0/24 is the lab side, 192.168.10.0/24 is my home network). Make sure both sides use different IP ranges to avoid conflicts. The manually configured route ensures that traffic destined for the other site’s network is forwarded through the tunnel. There’s also a netwatch script on both ends, which pings the other side’s tunnel endpoint to monitor the link. If the connection drops, the peer is toggled - this is needed because the link might not recover automatically.

On the peer list, you’ll notice the phone is marked as a “responder,” while the Site-to-Site peer isn’t. In simple terms, a “responder” is a device that doesn’t need to stay online all the time. For example, your phone can connect to the VPN on demand, so the server doesn’t have to constantly check its status. In contrast, a Site-to-Site tunnel is meant to be permanent and always up, so it’s actively monitored.

You can use any valid port for the WireGuard server.

- VLANs require more complex configuration. Assuming you’re a more knowledgeable network engineer, I won’t go through every detail the way I did for a non-VLAN setup.

  Main network (VLAN 10) functions like a non-VLAN setup: full access to network device management, no speed limits, and fasttracked connections. Guest network (VLAN 20), on the other hand, is restricted. It’s isolated from the main network and can’t access network devices - except R1, which provides essential services like DHCP, routing, and DNS. Access is limited per service port. Also, queues are set to 10 Mbps, as mentioned earlier, which means connection fasttracking must be disabled (see firewall rules). In this example, simple queues are used, but I highly recommend using queue trees with packet marking instead. It’s more complex to configure and requires better hardware to handle packet processing, but it’s worth it. More often than not, it’s not about the maximum speed the network can achieve -

it’s about traffic shaping and how well you manage it.

All access ports on network devices are configured with the main network VLAN ID - you can set the PVID per port as needed. Inter-VLAN configuration is flexible and can be set up however you want. You can create a DMZ zone for servers so they’re accessible from all VLANs, and for security, restrict access to specific ports on those servers. Alternatively, if you need to reach a printer in the office, you can pass multicast traffic using an mDNS repeater. It’s all up to you - RouterOS gives you that freedom.

I’ve divided the LAN group on R1 into three categories - full access, restricted, and combined. These groups are used in various places, so keep an eye out for them in the configurations. The full access group is for VLANs with no limits or restrictions. The restricted group is for isolated and limited VLANs, and the LAN group combines both of these. Make sure you assign VLANs to the appropriate groups, otherwise, you’ll run into issues.

CAPsMAN and Wi-Fi configuration also have noticeable differences. The most obvious one is that there are now two SSIDs - main and guest. Each SSID connects stations to its respective VLAN. Since the ax and ac lineup drivers differ slightly, their datapaths are also different. You can dynamically pass the VLAN ID to ax lineup APs, but this isn’t possible on ac devices - it requires manual configuration directly on the AP itself. First, provision the AP to CAPsMAN so that interfaces are created on the AP. Then, manually add the interfaces to the bridge on the AP side, just like in the example configuration. But before doing that, make sure the configuration is done on the server - for example, if you change the MAC address, it will trigger interface recreation on the AP, making the initially provisioned slave interfaces obsolete. This can create a mess on the AP side, so be careful.

APs are traditionally made for endpoint access to wired networks - they can’t be used as trunk ports to carry multiple VLAN IDs, at least not by default. WLAN interfaces are tagged with one specific PVID. To work around this limitation and extend the network with all VLANs over the PTP link, I created an EoIP tunnel. It’s a L2 logical tunnel over L3. The tunnel connects WBR1 to R1 over the local network and creates a logical interface on both devices, which is then added to the bridge. The main downside is that it changes how traffic flows from WBR1 access ports - now all traffic, even local traffic, must go through the tunnel to R1. Ports are set to “edge=yes”, because, as I understand it, BPDUs cause loop and it triggers port disablement. The tunnel itself adds overhead and disables any hardware offloading, so depending on your devices, it might impact WBR1’s

throughput. Also, VXLAN can be used and it can even be hardware offloaded on some models, but during testing, it showed out-of-order packet issues - that's why I went with EoIP as the solution. In small networks, it makes no difference, as both protocols do the same thing and are processed on the CPU anyway. On larger networks, you can set up tunnels between devices that support VXLAN hardware offloading, but always test packet flow quality. The out-of-order packet issue most likely is specific to this lab setup.

STP protocol is changed to MSTP, as it is the appropriate protocol for VLAN-aware networks. It's more complex, but also more versatile and highly scalable. When configuring, make sure the MSTP region configuration, including MST instances, is consistent across all affected bridges; otherwise, root election will break. The network is divided into two regions since EoIP breaks STP in this setup. Additionally, VLAN ID 2, used here for LTE passthrough, is assigned to a different root bridge because it's only used by LTE1 and R1.

In this configuration, every trunk port passes all tagged VLANs used on the network. This helps streamline configuration management and makes troubleshooting easier. But make sure you set a fixed PVID on access ports.

VLANs add complexity to the network, which means you might need better hardware. Make sure you understand the configuration requirements before you start planning, otherwise some parts of the network might underperform.

- I'd recommend learning Wireshark and capturing traffic on relevant ports, it'll help you better understand how systems work, troubleshoot issues, and keep things secure.
- Unnecessary IP services like API, FTP, and SSH have been disabled. Only Winbox and HTTP access are left enabled. I'd recommend turning off HTTP and sticking to Winbox only.
- As the L3 MTU is fixed on bridges, the L2 MTU stays at default values, which vary across models. This usually isn't an issue in simple networks, since typical setups don't significantly increase L2 frame sizes. Still, it's worth keeping in mind, especially as networks grow more complex and additional protocols that need larger frames are introduced.
- Set up a new admin account and make sure the RoMON and Wi-Fi passwords are updated. Always double-check that WireGuard keys are

unique.

- In this topology, I demonstrate how to connect network devices and encourage you to think about packet flow in the network. While network designs can vary depending on your requirements, the core principles remain the same. For example, if you don’t use a dedicated switch (SW1) and instead connect APs (AP1 and AP2) directly to the router (R1), the router becomes the core switch in this scenario. In such cases, you need to configure R1 as the STP root bridge and make sure other bridges point to it.
- Bandwidth Test is currently disabled - you can enable it if needed. Just keep in mind that to test real throughput on network devices, you need to run tests through them by setting up dedicated server and client endpoints, like with iperf3. Generating traffic uses CPU resources, and some packet flow configurations might be skipped, which could affect the results.
- FQ-CoDel queue type is set on WLAN interfaces and on simple queues, it's a more modern queuing option.
- Network flow was tested from every point in the topology to confirm performance expectations and ensure frames are delivered correctly - no sequencing issues, leaks, unneeded fragmentation, or other problems. Tests were run using iperf3 and the Bandwidth Test tool.
- The lab was built on RouterOS 7.20beta6.

Known issues, limitations and planned improvements

- Sniffing, torch, or any other packet inspection on LTE1 causes a loop. Currently, I don’t know the root cause - the configurations look fine, and all tests show no packet leaks across interfaces. If you need to inspect traffic flow, do it from R1 instead.
- The EoIP tunnel in the VLAN config creates a loop due to BPDUs traveling over it, so I had to set "edge=yes" on the logical interfaces at both ends of the tunnel. This breaks STP, as mentioned earlier, so I had to create a new STP region on WBR1.
- At first, I also had an interface toggle script running on LTE1 as a safety measure in case the LTE connection fails and doesn’t recover on its own. But setting up a script that toggles the interface every x minutes while the session is down turned out to be more complex than expected.
- A more advanced queue example is needed for complex setups and requirements.
- No IPv6
- Documentation is WIP.

Conclusion

I tried to create a simple, yet modern setup with stability and security in mind. No unnecessary configuration is included - every line has its purpose.

However, there are missing features, so don’t apply this setup blindly to production environments. For example, IPv6 isn’t configured yet (I’ll add it once my ISP supports it), and there’s no PPPoE setup, among other things.

I use these configurations in lab work and when designing real networks. As the saying goes, "Eat your own dog food." This lab is a reflection of my knowledge and experience. This particular setup was built, tested, and refined over the course of about a month.

I’ve uploaded everything to GitHub so I can tweak it, fix issues, and add features over time, plus, the community can track changes and give feedback.

Disclaimer: These configurations are my personal take on a basic setup. RouterOS is super flexible, you can go a different way depending on your needs or preferences. That said, some things are generally a bad idea, like disabling STP. Other parts are just preference: IP ranges (as long as they’re RFC 1918-compliant), VLAN design, whether to use RoMON, Wi-Fi settings, IPv6 setup, and so on.

Hope this helps and that you picked up something useful along the way.