Setting Up a Virtual Ubuntu Environment for the Course

Overview

This guide will help you set up an **Ubuntu 22.04+ VM** on **VirtualBox**, whether your **host machine** is running on **Intel** (x86) or **ARM** (e.g., macOS with Apple Silicon).

This VM will be used throughout the course for labs requiring kernel-level access (eBPF, perf, cgroups, etc.).

Step 1: Install VirtualBox and Prepare Ubuntu VM

1.1 Install VirtualBox

For Intel-based systems (most Windows/Linux PCs):

- Download and install VirtualBox from virtualbox.org
- Follow the installation instructions for your OS

For ARM-based systems (Apple Silicon macOS):

- VirtualBox support is limited on ARM
- **Recommended alternatives:**
  - **UTM** (free): mac.getutm.app
  - **Parallels Desktop** (paid, best performance)
  - **VMware Fusion** (free for personal use)

1.2 Download Ubuntu ISO

Download **Ubuntu 22.04 LTS** or **24.04 LTS**:

- **Intel/AMD hosts:** ubuntu.com/download/desktop
- **ARM hosts (Apple Silicon):** ubuntu.com/download/server/arm

1.3 Create the Virtual Machine

1. Open **VirtualBox** and click **New**
2. Set:
   - **Name:** `Ubuntu-OS-Course`
   - **Type:** Linux
   - **Version:** Ubuntu (64-bit)
3. **RAM:** 4GB minimum (8GB recommended)
4. **CPU:** 2 cores minimum
5. **Disk:** 25GB or larger

1.4 Install Ubuntu in the VM

1. Attach the Ubuntu ISO to the VM's optical drive
2. Start the VM and follow the installer
3. Set up your user account and password
4. After installation, remove the ISO and reboot

Step 2: Install Development Tools

Once Ubuntu is running, open a terminal and run:

2.1 Update the System

```
sudo apt update
sudo apt upgrade -y
```

2.2 Install Essential Tools

Minimum required for Week 1-3:

```
sudo apt install -y \
    build-essential \
    git \
    curl \
    wget \
    htop \
    strace \
    linux-tools-common \
    linux-tools-$(uname -r)
```

Additional tools for Week 4+ (eBPF):

```
sudo apt install -y \
    clang \
    llvm \
    libbpf-dev \
    libelf-dev \
    linux-headers-$(uname -r) \
    bpfcc-tools \
    bpftrace
```

Stress testing tools (Week 5+):

```
sudo apt install -y stress-ng fio
```

Python for data analysis:

```
sudo apt install -y python3 python3-pip
pip3 install matplotlib pandas numpy
```

Package Summary

Package	Purpose	When Needed
build-essential, gcc	C compilation	Week 1+
linux-tools	perf for profiling	Week 1+

strace	System call tracing	Week 1+
clang, llvm	eBPF compilation	Week 4+
bpftrace	eBPF tracing	Week 4+
stress-ng, fio	Workload generation	Week 5+

Step 3: Verify Your Environment

3.1 Check Compilation Tools

```
gcc --version
make --version
```

3.2 Check Perf

```
sudo perf stat ls
```

If this works without errors, `perf` is correctly installed.

3.3 Run the Environment Check Script

```
bash env_check.sh | tee lab0_env_check.txt
```

Review the output:

- `[OK]` = Good
- `[WARN]` = Fix later (OK for Week 1)
- `[ERROR]` = Fix now

Step 4: Troubleshooting

Permission Denied with perf

```
# Option 1: Use sudo
sudo perf stat ./program

# Option 2: Lower security level (your own VM only)
sudo sysctl kernel.perf_event_paranoid=1
```

Missing Kernel Headers

```
sudo apt install linux-headers-$(uname -r)
```

If your kernel was recently updated, reboot first, then re-run the command.

VM is Very Slow

- Increase RAM to 4GB or more
- Enable VT-x (Intel) or AMD-V (AMD) in BIOS settings
- Close unnecessary applications on host

VirtualBox Network Issues

- Default (NAT mode) should work for most cases
- If no internet, check VM Settings → Network → NAT

ARM Mac Issues

If you're on Apple Silicon (M1/M2/M3):

- VirtualBox support is limited
- Use **UTM** or **Parallels** instead
- Or use a cloud VM (AWS, GCP, etc.)

Step 5: Optional Enhancements

SSH Access for Remote Development

For easier access from VS Code on your host:

1. **Inside the VM**, install SSH server:

   ```
   sudo apt install -y openssh-server
   sudo systemctl enable --now ssh
   ```

2. **Configure VirtualBox port forwarding** (for NAT mode):

   - VM Settings → Network → Advanced → Port Forwarding
   - Add rule: Host Port `2222` → Guest Port `22`

3. **Connect from host:**

   ```
   ssh -p 2222 your_username@localhost
   ```

VS Code Remote Development

1. Install the **Remote – SSH** extension in VS Code
2. Add host entry in `~/.ssh/config`:

   ```
   Host os-course-vm
       HostName localhost
       Port 2222
       User your_username
   ```

3. Connect via VS Code's Remote Explorer

Shared Folders

To share files between host and VM:

1. Install VirtualBox Guest Additions inside the VM
2. Configure shared folder in VM Settings → Shared Folders
3. Mount inside VM:

   ```
   sudo mount -t vboxsf shared_folder_name /mnt/shared
   ```

Summary

After completing this guide, your VM should have:

Component	Status
Ubuntu 22.04+	✓ Installed
Build tools (gcc, make)	✓ Installed
Perf	✓ Working
Strace	✓ Working
eBPF tools (optional for Week 1)	✓ or pending

You are now ready for Lab 0!

Getting Help

If you encounter issues:

1. Check the troubleshooting section above
2. Search for the error message online
3. Ask during office hours or lab workshop
4. Post on course forum

Don't struggle alone — environment issues are common and fixable!