Ubuntu Docker VM Usage Guide

Quick Start

Starting Your Development Session

1. **Start the container** (if not already running):

```
./scripts/start.sh
```

2. **Connect to the container**:

```
./scripts/connect.sh
```

Or manually:

```
docker exec -it ubuntu-ai-vm bash
```

3. **Start developing**:

```
# Inside the container
cd /workspace
# Your development environment is ready
```

Ending Your Session

1. **Exit the container**:

```
exit
```

2. **Stop the container** (optional - you can leave it running):

```
./scripts/stop.sh
```

Development Workflow

Example Python Project

```
# Connect to container
./scripts/connect.sh

# Navigate to workspace
cd /workspace

# Create a new project
mkdir my-python-project
cd my-python-project

# Create a virtual environment
python3 -m venv venv
source venv/bin/activate

# Install packages
pip install flask pytest

# Start coding
vim app.py  # or nano app.py
```

Example Node.js Project

```
# Connect to container
./scripts/connect.sh

# Navigate to workspace
cd /workspace

# Create a new project
mkdir my-node-project
cd my-node-project

# Initialize npm project
npm init -y

# Install packages
npm install express jest

# Start coding
vim index.js  # or nano index.js
```

Installing AI Coding Assistants

You can install various AI coding tools inside the container:

```
# Install Claude Code CLI
curl -sL https://github.com/anthropics/claude-
code/releases/latest/download/claude-code_linux_arm64.tar.gz | tar xz
```

```
sudo mv claude-code /usr/local/bin/

# Install GitHub Copilot CLI
npm install -g @githubnext/github-copilot-cli

# Install other AI tools as needed
pip3 install aider-chat  # Aider
pip3 install gpt-engineer  # GPT Engineer
```

File Management

Accessing Your Files

All your work is saved in the `/workspace` directory inside the container, which persists between sessions.

From inside the container:

```
cd /workspace
ls -la
```

From your host machine:

```
# View container files
docker exec ubuntu-ai-vm ls -la /workspace

# Copy files TO the container
docker cp myfile.txt ubuntu-ai-vm:/workspace/

# Copy files FROM the container
docker cp ubuntu-ai-vm:/workspace/myfile.txt ./

# Copy entire directories
docker cp my-project ubuntu-ai-vm:/workspace/
docker cp ubuntu-ai-vm:/workspace/my-project ./
```

Backing Up Your Work

To backup your workspace:

```
# Create a backup directory
mkdir -p backups

# Backup the entire workspace
docker exec ubuntu-ai-vm tar -czf - -C /workspace . >
backups/workspace-$(date +%Y%m%d-%H%M%S).tar.gz
```

To restore from backup:

```
# Restore to workspace
docker exec -i ubuntu-ai-vm tar -xzf - -C /workspace < backups/workspace-
20240706-123456.tar.gz
```

Advanced Usage

Installing Additional Software

The `aiuser` has limited sudo access for package management:

```
# Inside the container
sudo apt-get update
sudo apt-get install package-name

# For Python packages
pip3 install package-name

# For Node.js packages
npm install package-name
```

Working with Git

Git is pre-installed in the container:

```
# Configure git (inside container)
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"

# Clone repositories
cd /workspace
git clone https://github.com/username/repo.git

# Work with your repos normally
cd repo
git add .
git commit -m "Update: implemented new feature"
git push
```

Using Different Terminals

You can open multiple terminal sessions:

```

```

```
# Terminal 1: Run your application
./scripts/connect.sh
python3 app.py

# Terminal 2: Edit code
./scripts/connect.sh
vim app.py  # or use your preferred editor

# Terminal 3: Run tests
./scripts/connect.sh
pytest
```

Environment Variables

Set persistent environment variables:

```
# Inside container
echo 'export MY_VAR="value"' >> ~/.bashrc
source ~/.bashrc
```

Troubleshooting

Container Won't Start

Error: Docker daemon not running

```
# On macOS
open -a Docker
# Wait 30 seconds for Docker to start
./scripts/start.sh
```

Error: Port already in use

```
# Stop all containers
docker stop $(docker ps -aq)
# Try starting again
./scripts/start.sh
```

Container Starts but Can't Connect

Check if container is running:

```
docker ps | grep ubuntu-ai-vm
```

Check container logs:

```
docker logs ubuntu-ai-vm
```

Restart the container:

```
./scripts/stop.sh
./scripts/start.sh
```

Internet Connectivity Issues

Test connectivity:

```
docker exec ubuntu-ai-vm curl -I https://www.google.com
```

Check DNS:

```
docker exec ubuntu-ai-vm nslookup google.com
```

Restart Docker network:

```
docker network prune -f
./scripts/stop.sh
./scripts/start.sh
```

Development Tool Issues

Python package not found:

```
# Update pip and reinstall
pip3 install --upgrade pip
pip3 install package-name
```

Node package issues:

```
# Clear npm cache
npm cache clean --force
# Reinstall dependencies
```

```
rm -rf node_modules package-lock.json
npm install
```

Permission errors:

```
# For global npm packages
npm config set prefix ~/.npm-global
echo 'export PATH=~/.npm-global/bin:$PATH' >> ~/.bashrc
source ~/.bashrc
```

Storage Issues

Check available space:

```
docker exec ubuntu-ai-vm df -h /workspace
```

Clean up Docker space:

```
# Remove unused images and containers
docker system prune -a
# Clean build cache
docker builder prune
```

Resize storage (if using sparse file):

```
./scripts/resize-volume.sh 40  # Resize to 40GB
```

Permission Issues

Can't write to workspace:

```
# Fix permissions from inside container
sudo chown -R aiuser:aiuser /workspace
```

Can't install packages:

```
# The aiuser can only use apt-get/apt
sudo apt-get install package-name
# For other sudo commands, you'll need to modify the Dockerfile
```

Performance Issues

Container running slowly:

```
# Check resource usage
docker stats ubuntu-ai-vm

# Restart with more resources
# Edit docker-compose.yml and increase:
# - cpus: '4'  # Increase CPU
# - memory: 8G  # Increase RAM
./scripts/stop.sh
./scripts/start.sh
```

Best Practices

1. **Regular Backups**: Backup your workspace regularly
2. **Git Commits**: Commit your code changes frequently
3. **Container Hygiene**: Stop the container when not in use for extended periods
4. **API Keys**: Never commit API keys to git; use environment variables
5. **Updates**: Periodically update packages inside the container

Useful Aliases

Add these to your host machine's shell config for convenience:

```
# Add to ~/.bashrc or ~/.zshrc
alias ai-start='cd ~/ubuntu-docker-vm && ./scripts/start.sh'
alias ai-connect='cd ~/ubuntu-docker-vm && ./scripts/connect.sh'
alias ai-stop='cd ~/ubuntu-docker-vm && ./scripts/stop.sh'
alias ai-backup='cd ~/ubuntu-docker-vm && docker exec ubuntu-ai-vm tar -
czf - -C /workspace . > backups/workspace-$(date +%Y%m%d-%H%M%S).tar.gz'
```

Emergency Recovery

If something goes wrong:

1. **Save your work** (if possible):

   ```
   docker cp ubuntu-ai-vm:/workspace ./workspace-emergency-backup
   ```

2. **Use the rollback script**:

   ```
   ./rollback.sh
   ```

3. **Start fresh**:

```
./scripts/start.sh
```

4. **Restore your work**:

```
docker cp ./workspace-emergency-backup/. ubuntu-ai-vm:/workspace/
```

Quick Command Reference

Task	Command
Start container	`./scripts/start.sh`
Connect to container	`./scripts/connect.sh`
Stop container	`./scripts/stop.sh`
Check status	`docker ps | grep ubuntu-ai-vm`
View logs	`docker logs ubuntu-ai-vm`
Copy file to container	`docker cp file.txt ubuntu-ai-vm:/workspace/`
Copy file from container	`docker cp ubuntu-ai-vm:/workspace/file.txt ./`
Backup workspace	`docker exec ubuntu-ai-vm tar -czf - -C /workspace . > backup.tar.gz`
Start Python shell	`python3` (inside container)
Install Python package	`pip3 install package` (inside container)
Install system package	`sudo apt-get install package` (inside container)

Remember: All your coding work should be done inside the `/workspace` directory to ensure it persists between container restarts!