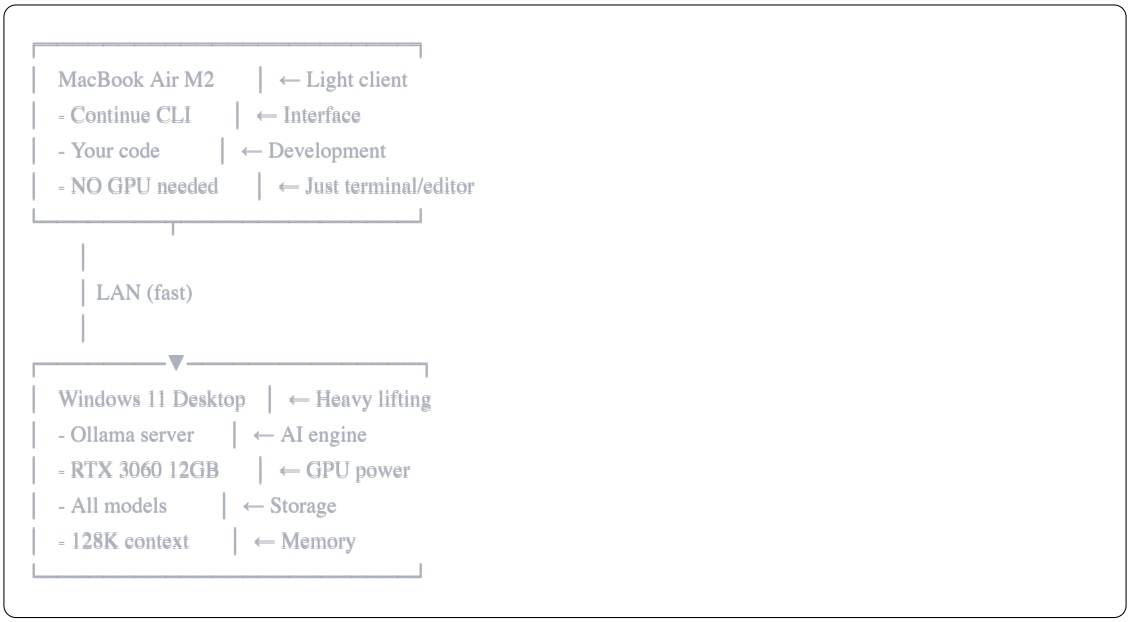
RTX 3060 Setup Guide - Complete AI Coding Workflow

Your Architecture (Optimal)



Phase 1: Hardware Installation & Driver Setup

Step 1: Install RTX 3060

1. Shut down Windows 11 machine
2. Install RTX 3060 in PCIe slot
3. Connect power cables (if required)
4. Boot up

Step 2: Install/Update NVIDIA Drivers

```powershell
# Download latest drivers from:
# https://www.nvidia.com/Download/index.aspx
# Search: RTX 3060, Windows 11

# Or use GeForce Experience (recommended)
# https://www.nvidia.com/en-us/geforce/geforce-experience/

# After install, verify:
nvidia-smi
# Should show RTX 3060 with 12GB VRAM
```

Step 3: Install CUDA Toolkit (if not present)

powershell

```
# Download from:
# https://developer.nvidia.com/cuda-downloads
# Choose: Windows, x86_64, 11, exe (local)

# Verify installation:
nvcc --version
# Should show CUDA version 12.x
```

Phase 2: Clean Up Old Models (Optional)

Models to DELETE (free up space):

bash

```
# On Windows 11 Ollama server:
ollama rm tinyllama:latest          # Too weak
ollama rm qwen2.5:1.5b              # Redundant
ollama rm gemma2:2b                 # Redundant
ollama rm qwen2.5-coder:1.5b        # Upgrading to 14B
ollama rm notus:latest              # Outdated
ollama rm alfred:latest             # Niche use
ollama rm firefunction-v2:latest    # Redundant
ollama rm gemma3:1b                 # Too small
```

Models to KEEP:

bash

```
# Keep these (still useful):
phi3:mini                 # Fast 3.8B for quick queries
gemma3:4b                 # Lightweight backup
mistral:latest            # Good 7B model
llava:13b                 # Vision model (will work now!)
deepseek-r1:latest        # Reasoning (if 8B or 14B)
qwen3-coder:latest        # Check size, keep if 14B or less
```

Phase 3: Download Essential Models

Priority 1: Core Coding Models (Download First)

bash

```
# BEST: Coding with large context (PRIMARY MODEL)
ollama pull qwen2.5-coder:14b
# Size: ~9GB, VRAM: ~9GB, Speed: 25-30 tok/s, Context: 32K

# BEST: Coding with huge context (ALTERNATIVE PRIMARY)
ollama pull deepseek-coder-v2:16b
# Size: ~9GB, VRAM: ~10GB, Speed: 20-25 tok/s, Context: 128K

# BACKUP: Fast coding model
ollama pull qwen2.5-coder:7b
# Size: ~4.7GB, VRAM: ~5GB, Speed: 40-45 tok/s, Context: 32K
```

Priority 2: General Chat & Reasoning

bash

```
# General chat/writing
ollama pull qwen2.5:14b
# Size: ~9GB, VRAM: ~9GB, Speed: 25-30 tok/s, Context: 32K

# Complex reasoning (upgrade from current)
ollama pull deepseek-r1:14b
# Size: ~9GB, VRAM: ~9GB, Speed: 20-25 tok/s, Context: 8K

# Alternative: Excellent reasoning
ollama pull phi4:14b
# Size: ~9GB, VRAM: ~9GB, Speed: 25-30 tok/s, Context: 16K
```

Priority 3: Specialized (Optional)

bash

```
# If you do Python-heavy work
ollama pull codellama:13b-python
# Size: ~7.4GB, VRAM: ~8GB, Speed: 25-30 tok/s

# If you want vision capabilities (already have llava:13b)
# But this is better:
ollama pull llava-llama3:8b
# Size: ~5.5GB, VRAM: ~6GB, Speed: 30-35 tok/s (faster alternative)

# Large context general model
ollama pull codestral:22b
# Size: ~13GB, VRAM: ~11GB, Speed: 15-20 tok/s, Context: 128K
# NOTE: Might be tight on 12GB VRAM
```

Phase 4: Configure Continue CLI (MacBook Air M2)

Location: `~/.continue/config.yaml`

Complete Optimal Configuration:

yaml

```yaml
name: My Config (RTX 3060 Optimized)
version: 1.0.0
schema: v1

models:
  # ============================================
  # PRIMARY MODELS (Daily Use)
  # ============================================

  - name: Qwen2.5 Coder 14B (Default)
    provider: ollama
    model: "qwen2.5-coder:14b"
    apiBase: https://ollama.ldmathes.cc
    baseUrl: https://ollama.ldmathes.cc
    contextLength: 32768
    default: true  # ← YOUR DEFAULT MODEL
    roles:
      - chat
      - edit
    capabilities:
      - tool_use
    requestOptions:
      timeout: 300000
      keepAlive: 300000

  - name: DeepSeek Coder V2 16B (Large Context)
    provider: ollama
    model: "deepseek-coder-v2:16b"
    apiBase: https://ollama.ldmathes.cc
    baseUrl: https://ollama.ldmathes.cc
    contextLength: 131072  # 128K tokens!
    roles:
      - chat
      - edit
    capabilities:
      - tool_use
    requestOptions:
      timeout: 600000  # Longer for big context
      keepAlive: 600000

  - name: Qwen Fast 7B
    provider: ollama
    model: "qwen2.5-coder:7b"
    apiBase: https://ollama.ldmathes.cc
    baseUrl: https://ollama.ldmathes.cc
    contextLength: 32768
    roles:
      - chat
    requestOptions:
      timeout: 300000
      keepAlive: 300000

  # ============================================
  # AUTOCOMPLETE (Fast, Small)
  # ============================================

  - name: Qwen Autocomplete
    provider: ollama
```

```
  provider: ollama
  model: "qwen2.5-coder:1.5b"
  apiBase: https://ollama.ldmathes.cc
  baseUrl: https://ollama.ldmathes.cc
  contextLength: 32768
  roles:
    - autocomplete
  requestOptions:
    timeout: 60000
    keepAlive: 60000

# ========================================
# REASONING & GENERAL USE
# ========================================

- name: DeepSeek R1 14B (Reasoning)
  provider: ollama
  model: "deepseek-r1:14b"
  apiBase: https://ollama.ldmathes.cc
  baseUrl: https://ollama.ldmathes.cc
  contextLength: 8192
  roles:
    - chat
  requestOptions:
    timeout: 300000
    keepAlive: 300000

- name: Qwen2.5 14B (General Chat)
  provider: ollama
  model: "qwen2.5:14b"
  apiBase: https://ollama.ldmathes.cc
  baseUrl: https://ollama.ldmathes.cc
  contextLength: 32768
  roles:
    - chat
  requestOptions:
    timeout: 300000
    keepAlive: 300000

- name: Phi4 14B (Alternative Reasoning)
  provider: ollama
  model: "phi4:14b"
  apiBase: https://ollama.ldmathes.cc
  baseUrl: https://ollama.ldmathes.cc
  contextLength: 16384
  roles:
    - chat
  requestOptions:
    timeout: 300000
    keepAlive: 300000

# ========================================
# BACKUP MODELS (Keep but not default)
# ========================================

- name: Phi3 Mini (Fast Backup)
  provider: ollama
  model: "phi3:mini"
  apiBase: https://ollama.ldmathes.cc
  baseUrl: https://ollama.ldmathes.cc
```

```
    contextLength: 131072  # 128K context
    roles:
      - chat
    requestOptions:
      timeout: 300000
      keepAlive: 300000

  - name: Mistral 7B
    provider: ollama
    model: "mistral:latest"
    apiBase: https://ollama.ldmathes.cc
    baseUrl: https://ollama.ldmathes.cc
    contextLength: 32768
    roles:
      - chat
    requestOptions:
      timeout: 300000
      keepAlive: 300000

  - name: LLaVA Vision 13B
    provider: ollama
    model: "llava:13b"
    apiBase: https://ollama.ldmathes.cc
    baseUrl: https://ollama.ldmathes.cc
    contextLength: 4096
    roles:
      - chat
    requestOptions:
      timeout: 300000
      keepAlive: 300000

  # ========================================
  # CLOUD MODELS (Emergency Use Only)
  # ========================================

  - name: Kimi K2 (Cloud - Limited Quota)
    provider: ollama
    model: "kimi-k2-thinking:cloud"
    apiBase: https://ollama.ldmathes.cc
    baseUrl: https://ollama.ldmathes.cc
    roles:
      - chat
    requestOptions:
      timeout: 600000
      keepAlive: 600000

  - name: DeepSeek V3.1 (Cloud - Limited Quota)
    provider: ollama
    model: "deepseek-v3.1:671b-cloud"
    apiBase: https://ollama.ldmathes.cc
    baseUrl: https://ollama.ldmathes.cc
    roles:
      - chat
    requestOptions:
      timeout: 600000
      keepAlive: 600000
```

Phase 5: Test Your Setup

Test 1: Verify GPU is being used

```bash
# On Windows 11 (during inference):
nvidia-smi

# Should show:
# - GPU utilization: 80-100%
# - Memory usage: 8-10GB (for 14B models)
# - Temperature: 60-75°C
```

Test 2: Speed test from MacBook

```bash
# On MacBook Air M2:
cn

> Write a Python function to calculate fibonacci numbers

# Observe:
# - Response should stream quickly (20-30 tokens/sec)
# - Total response time: 3-6 seconds
# - Compare to old: 20-40 seconds
```

Test 3: Context test

```bash
cn

> @ui_components.py @keystroke_handler.py @browser_controller.py @inject.py @names.json
> Analyze this codebase architecture

# Should complete without timeout
# Can follow up with 20+ more questions
```

Test 4: Model switching

```bash
cn

> Use DeepSeek Coder V2 16B (Large Context) for this question:
> [complex multi-file question]

# Should switch models and respond
```

Phase 6: Optimal Usage Patterns

Daily Coding Workflow

```bash
# Morning: Start with default model
cn
> @ui_components.py
> Add new feature X

# Iterative development (unlimited)
> That works, now add Y
> Refactor this section
> Add error handling
> Generate tests
# ... 20+ exchanges, no quota limits

# Complex architecture question
> Use DeepSeek Coder V2 16B (Large Context):
> @all-5-files
> How should I restructure this?

# Quick syntax question
> Use Qwen Fast 7B:
> What's the Python syntax for X?
```

Model Selection Guide

Task	Model	Why
Daily coding	Qwen2.5 Coder 14B	Best balance
Long conversations	DeepSeek Coder V2 16B	128K context
Quick questions	Qwen Fast 7B	Speed
Complex debugging	DeepSeek R1 14B	Reasoning
General writing	Qwen2.5 14B	Not code-focused
Vision tasks	LLaVA 13B	Image understanding
Emergency/stuck	Cloud models	Save for last resort

Phase 7: Performance Benchmarks

Expected Performance (RTX 3060 12GB)

Model	VRAM Usage	Speed	Quality	Context
qwen2.5-coder:14b	9GB	25-30 tok/s	Excellent	32K
deepseek-coder-v2:16b	10GB	20-25 tok/s	Excellent	128K
qwen2.5-coder:7b	5GB	40-45 tok/s	Very Good	32K
deepseek-r1:14b	9GB	20-25 tok/s	Excellent	8K
phi3:mini	3GB	50-60 tok/s	Good	128K

vs Current Setup (GT 1030 CPU)

Metric	GT 1030 (CPU)	RTX 3060 (GPU)	Improvement
Speed	2-5 tok/s	25-30 tok/s	**10x faster**
Response time	30-60 sec	3-6 sec	**10x faster**
Model quality	1.5-4B	14-16B	**4x larger**
VRAM capacity	1GB (unused)	12GB	**12x more**
Context	32K	128K	**4x larger**

Phase 8: Troubleshooting

Problem: Models run on CPU instead of GPU

```powershell
# Check CUDA is working:
nvidia-smi

# Restart Ollama:
# (In Services or Task Manager, restart Ollama)

# Check Ollama sees GPU:
ollama list
# Look for GPU indicator in output
```

Problem: Out of VRAM errors

```bash
# Check current VRAM usage:
nvidia-smi

# Stop other GPU applications
# Use smaller model:
ollama pull qwen2.5-coder:7b

# Or use quantized version:
ollama pull qwen2.5-coder:14b-q4
```

Problem: Slow responses from MacBook

```bash
# Test network latency:
ping ollama.ldmathes.cc

# Should be < 20ms
# If higher, check WiFi/network

# Test direct to Windows:
curl https://ollama.ldmathes.cc/api/tags

# Should respond quickly
```

Problem: Continue can't connect

```bash
# On MacBook, verify config:
cat ~/.continue/config.yaml | grep apiBase

# Should show: https://ollama.ldmathes.cc

# Test from MacBook:
curl https://ollama.ldmathes.cc/api/tags

# Should return JSON with model list
```

Phase 9: Future Optimizations

Option 1: Run Ollama locally on MacBook (Alternative)

```bash
# If you want everything local on M2:
# (Not recommended - desktop GPU is better)

# Install Ollama on MacBook:
brew install ollama

# Download smaller models:
ollama pull qwen2.5-coder:7b
ollama pull phi3:mini

# Update config.yaml:
apiBase: http://localhost:11434

# Trade-off:
# + No network dependency
# - Slower (M2 vs RTX 3060)
# - Battery drain
# - Thermal throttling
```

Option 2: Increase Context Further

```bash
# Create custom 256K context model:
# (Advanced - may reduce quality)

cat > Modelfile.qwen-256k << 'EOF'
FROM qwen2.5-coder:14b
PARAMETER num_ctx 262144
PARAMETER rope_frequency_scale 0.125
PARAMETER rope_frequency_base 1000000
EOF

ollama create qwen-coder-256k -f Modelfile.qwen-256k

# Add to config.yaml:
- name: Qwen Coder 256K (Experimental)
  model: "qwen-coder-256k"
  contextLength: 262144
```

Option 3: Add More Specialized Models

```bash
# If you work with specific languages:
ollama pull codellama:13b-python
ollama pull starcoder2:15b

# If you need multimodal:
ollama pull llava-llama3:13b
ollama pull bakllava:latest

# If you need larger models (tight fit):
ollama pull qwen2.5-coder:32b  # Will use 11-12GB
```

Phase 10: Maintenance & Updates

Weekly Tasks

```bash
# Check for Ollama updates:
# Download latest from ollama.ai

# Update models:
ollama pull qwen2.5-coder:14b
ollama pull deepseek-coder-v2:16b

# Clean up old versions:
ollama list
ollama rm old-model:old-version
```

Monthly Tasks

```bash
# Check NVIDIA driver updates
# Update CUDA if needed

# Review model usage:
# Remove unused models
# Add new released models

# Backup config:
cp ~/.continue/config.yaml ~/.continue/config.yaml.backup
```

Quick Reference Card

Essential Commands (MacBook)

```bash
cn                    # Start Continue CLI
^D                    # Exit session
@filename.py          # Load file context
@codebase             # Search all files
```

Essential Commands (Windows)

```bash
ollama list              # Show all models
ollama pull model:tag    # Download model
ollama rm model:tag      # Remove model
ollama run model:tag     # Test model
nvidia-smi               # Check GPU usage
```

Model Selection Quick Guide

```bash
# Default: qwen2.5-coder:14b
# Large context: deepseek-coder-v2:16b
# Speed: qwen2.5-coder:7b
# Reasoning: deepseek-r1:14b
# Backup: phi3:mini
```

Summary: The New Workflow

Before (GT 1030):

- Speed: 2-5 tok/s (painful)
- Models: 1.5-4B (limited)
- Context: 32K (okay)
- Cloud dependency: High (quotas)

After (RTX 3060):

- Speed: 25-30 tok/s (smooth)
- Models: 14-16B (excellent)
- Context: 128K (huge)
- Cloud dependency: None (unlimited)

Result:

- 10x faster responses
- 4x better model quality
- 4x larger context
- Unlimited usage
- Professional-grade local AI coding

Installation Checklist

- [ ] Install RTX 3060 hardware
- [ ] Install NVIDIA drivers
- [ ] Install CUDA toolkit
- [ ] Test nvidia-smi shows GPU
- [ ] Delete old small models
- [ ] Download qwen2.5-coder:14b
- [ ] Download deepseek-coder-v2:16b
- [ ] Download deepseek-r1:14b
- [ ] Download qwen2.5:14b
- [ ] Update config.yaml on MacBook
- [ ] Test cn connects and shows new models
- [ ] Run speed test
- [ ] Run context test
- [ ] Verify GPU usage with nvidia-smi
- [ ] Save this document for reference

You're ready to code with unlimited local AI power! 🚀