High-Performance Development Environment Setup

Optimized for ThinkPad P1 Gen 7 - 64GB RAM, 22-core Intel Ultra 7 165H

🚀 Phase 1: Foundation Setup

1. Development Tools Installation

bash

```bash
# Essential development packages
sudo dnf groupinstall "Development Tools" "Development Libraries"
sudo dnf install -y \
    git git-lfs \
    neovim code \
    docker podman \
    kubernetes kubectl helm \
    nodejs npm yarn \
    python3 python3-pip python3-venv \
    golang rust cargo \
    java-17-openjdk maven gradle \
    postgresql-server redis \
    nginx \
    tmux zellij \
    btop htop \
    jq yq \
    terraform \
    ansible

# Modern CLI tools for your .zshrc
sudo dnf install -y \
    bat exa ripgrep fd-find \
    procs dust duf \
    git-delta fzf \
    zoxide starship
```

2. Container Platform Setup (Leverage that 64GB RAM!)

bash

```bash
# Configure Podman for rootless containers
echo 'unqualified-search-registries = ["docker.io"]' | sudo tee -a /etc/containers/regi

# Increase container limits for your massive RAM
mkdir -p ~/.config/containers
cat > ~/.config/containers/storage.conf << 'EOF'
[storage]
driver = "overlay"
runroot = "/run/user/1000/containers"
graphroot = "/home/$USER/.local/share/containers/storage"

[storage.options]
size = "100G"
EOF

# Set resource limits to use your full potential
echo "vm.max_map_count=262144" | sudo tee -a /etc/sysctl.conf
sudo sysctl -w vm.max_map_count=262144
```

🎯 Phase 2: Multi-Environment Development Setup

1. Local Kubernetes Development

bash

```bash
# Install K3d for local Kubernetes (lightweight, perfect for development)
curl -s https://raw.githubusercontent.com/k3d-io/k3d/main/install.sh | bash

# Create a powerful local cluster using your resources
k3d cluster create dev-cluster \
    --agents 3 \
    --memory 16g \
    --cpus 8 \
    --api-port 6443 \
    --port 8080:80@loadbalancer \
    --port 8443:443@loadbalancer

# Install Kubernetes dashboard
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/depl
```

2. Development Databases (Containerized)

bash

```
# PostgreSQL for serious applications
podman run -d \
    --name postgres-dev \
    -e POSTGRES_PASSWORD=devpass \
    -e POSTGRES_DB=devdb \
    -p 5432:5432 \
    -v postgres-data:/var/lib/postgresql/data \
    postgres:15

# Redis for caching/sessions
podman run -d \
    --name redis-dev \
    -p 6379:6379 \
    redis:7-alpine

# MongoDB for NoSQL projects
podman run -d \
    --name mongo-dev \
    -e MONGO_INITDB_ROOT_USERNAME=admin \
    -e MONGO_INITDB_ROOT_PASSWORD=devpass \
    -p 27017:27017 \
    -v mongo-data:/data/db \
    mongo:6

# ClickHouse for analytics (your RAM can handle it!)
podman run -d \
    --name clickhouse-dev \
    -p 8123:8123 \
    -p 9000:9000 \
    --ulimit nofile=262144:262144 \
    -v clickhouse-data:/var/lib/clickhouse \
    clickhouse/clickhouse-server:latest
```

3. Message Queues & Streaming

bash

```bash
# Apache Kafka (memory-intensive, perfect for your setup)
podman run -d \
    --name zookeeper \
    -p 2181:2181 \
    -e ZOOKEEPER_CLIENT_PORT=2181 \
    confluentinc/cp-zookeeper:latest

podman run -d \
    --name kafka \
    -p 9092:9092 \
    -e KAFKA_ZOOKEEPER_CONNECT=localhost:2181 \
    -e KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://localhost:9092 \
    -e KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1 \
    -e KAFKA_HEAP_OPTS="-Xmx2G -Xms2G" \
    confluentinc/cp-kafka:latest

# RabbitMQ for traditional messaging
podman run -d \
    --name rabbitmq-dev \
    -p 5672:5672 \
    -p 15672:15672 \
    -e RABBITMQ_DEFAULT_USER=admin \
    -e RABBITMQ_DEFAULT_PASS=devpass \
    rabbitmq:3-management
```

🧠 Phase 3: AI/ML Development Environment

1. Python AI/ML Stack

bash

```bash
# Create dedicated Python environment for ML
python3 -m venv ~/venv/ml
source ~/venv/ml/bin/activate

# Install core ML libraries (your 64GB RAM can handle large models!)
pip install \
    torch torchvision torchaudio \
    tensorflow \
    scikit-learn pandas numpy \
    jupyter jupyterlab \
    transformers datasets \
    langchain openai \
    streamlit \
    mlflow wandb \
    dvc

# Start Jupyter Lab with increased resources
echo "c.NotebookApp.max_buffer_size = 2**28" > ~/.jupyter/jupyter_notebook_config.py
```

2. Local AI Model Serving

bash

```bash
# Ollama for local LLM serving (leveraging your hardware)
curl -fsSL https://ollama.com/install.sh | sh

# Start Ollama and pull models
systemctl --user enable ollama
systemctl --user start ollama

# Pull some models (your RAM can handle multiple large models)
ollama pull llama2:13b
ollama pull codellama:34b
ollama pull mistral:7b
```

🛠️ Phase 4: Development Tool Configuration

1. VS Code with Powerhouse Extensions

bash

```bash
# Install VS Code extensions for multi-language development
code --install-extension ms-python.python
code --install-extension golang.go
code --install-extension rust-lang.rust-analyzer
code --install-extension ms-vscode.vscode-typescript-next
code --install-extension ms-kubernetes-tools.vscode-kubernetes-tools
code --install-extension ms-azuretools.vscode-docker
code --install-extension hashicorp.terraform
code --install-extension redhat.ansible
code --install-extension ms-toolsai.jupyter
code --install-extension github.copilot

# Configure VS Code for your hardware
mkdir -p ~/.config/Code/User
cat > ~/.config/Code/User/settings.json << 'EOF'
{
    "files.watcherExclude": {
        "**/node_modules/**": true
    },
    "search.maxResults": 50000,
    "terminal.integrated.profiles.linux": {
        "zsh": {
            "path": "/usr/bin/zsh"
        }
    },
    "terminal.integrated.defaultProfile.linux": "zsh",
    "python.defaultInterpreterPath": "~/venv/ml/bin/python",
    "go.toolsManagement.autoUpdate": true,
    "rust-analyzer.server.path": "rust-analyzer",
    "kubernetes.kubectl-path.linux": "/usr/bin/kubectl"
}
EOF
```

2. Neovim Configuration (For Terminal Power Users)

bash

```bash
# Install LazyVim (modern Neovim distribution)
git clone https://github.com/LazyVim/starter ~/.config/nvim
rm -rf ~/.config/nvim/.git

# Install language servers for multi-language support
sudo dnf install -y \
    nodejs npm \
    python3-pip \
    rust-analyzer \
    gopls
```

🔄 Phase 5: Workflow Automation

1. Development Environment Manager

bash

```bash
# Create environment control script
cat > ~/bin/dev-env << 'EOF'
#!/bin/bash

case $1 in
    start)
        echo "🚀 Starting development environment..."
        systemctl --user start podman
        podman start postgres-dev redis-dev mongo-dev clickhouse-dev rabbitmq-dev 2>/de
        k3d cluster start dev-cluster 2>/dev/null
        systemctl --user start ollama
        echo "✅ Development environment ready!"
        ;;
    stop)
        echo "🛑 Stopping development environment..."
        podman stop postgres-dev redis-dev mongo-dev clickhouse-dev rabbitmq-dev 2>/dev
        k3d cluster stop dev-cluster 2>/dev/null
        systemctl --user stop ollama
        echo "✅ Development environment stopped!"
        ;;
    status)
        echo "📊 Development Environment Status:"
        echo "Containers:"
        podman ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"
        echo -e "\nKubernetes:"
        k3d cluster list
        echo -e "\nSystem Resources:"
        free -h
        ;;
    restart)
        $0 stop
        sleep 5
        $0 start
        ;;
    *)
        echo "Usage: dev-env {start|stop|status|restart}"
        ;;
esac
EOF

chmod +x ~/bin/dev-env
```

2. Resource Monitoring Dashboard

bash

```
# Install modern system monitoring
sudo dnf install -y btop bandwhich

# Create monitoring script
cat > ~/bin/monitor << 'EOF'
#!/bin/bash
# Multi-pane monitoring for development

tmux new-session -d -s monitor
tmux split-window -h
tmux split-window -v
tmux select-pane -t 0
tmux split-window -v

tmux send-keys -t 0 'btop' Enter
tmux send-keys -t 1 'watch -n 2 "podman ps --format \"table {{.Names}}\t{{.Status}}\t{{
tmux send-keys -t 2 'watch -n 5 "kubectl get pods -A"' Enter
tmux send-keys -t 3 'journalctl -f -u podman' Enter

tmux attach-session -t monitor
EOF

chmod +x ~/bin/monitor
```

🎮 Phase 6: Language-Specific Optimizations

1. Go Development (Optimized for Fast Compilation)

bash

```
# Configure Go for your multicore CPU
echo 'export GOMAXPROCS=22' >> ~/.zshrc
echo 'export GOCACHE=~/.cache/go-build' >> ~/.zshrc

# Install Go tools
go install golang.org/x/tools/gopls@latest
go install github.com/golangci/golangci-lint/cmd/golangci-lint@latest
go install github.com/air-verse/air@latest
```

2. Rust Development (Leverage Parallel Compilation)

bash

```bash
# Configure Rust for parallel builds
echo 'export CARGO_BUILD_JOBS=22' >> ~/.zshrc
echo 'export RUSTC_WRAPPER=sccache' >> ~/.zshrc

# Install sccache for compilation caching
cargo install sccache
```

3. Node.js Development (Multiple Versions)

bash

```bash
# Install Node Version Manager
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash

# Install multiple Node versions for different projects
nvm install --lts
nvm install 18
nvm install 20
nvm use --lts

# Configure npm for your setup
npm config set cache ~/.npm-cache
npm config set maxsockets 50
```

🚀 Phase 7: Performance Testing Environment

1. Load Testing Tools

bash

```bash
# Install performance testing tools
sudo dnf install -y apache-bench siege
cargo install drill
go install github.com/tsenart/vegeta@latest

# Install k6 for modern load testing
sudo dnf install xz
curl -s https://github.com/grafana/k6/releases/download/v0.47.0/k6-v0.47.0-linux-amd64.
sudo mv k6-v0.47.0-linux-amd64/k6 /usr/local/bin/
```

2. Monitoring Stack (Prometheus + Grafana)

bash

```
# Prometheus for metrics
podman run -d \
    --name prometheus \
    -p 9090:9090 \
    -v prometheus-data:/prometheus \
    prom/prometheus:latest

# Grafana for visualization
podman run -d \
    --name grafana \
    -p 3000:3000 \
    -e GF_SECURITY_ADMIN_PASSWORD=admin \
    -v grafana-data:/var/lib/grafana \
    grafana/grafana:latest
```

🎯 Phase 8: Quick Start Commands

Add these to your `.zshrc`:

bash

```bash
# Development environment shortcuts
alias dev-start='~/bin/dev-env start'
alias dev-stop='~/bin/dev-env stop'
alias dev-status='~/bin/dev-env status'
alias dev-monitor='~/bin/monitor'

# Quick service connections
alias pg-connect='psql postgresql://postgres:devpass@localhost:5432/devdb'
alias redis-cli='redis-cli -h localhost -p 6379'
alias mongo-connect='mongosh "mongodb://admin:devpass@localhost:27017/admin"'

# Container management
alias containers='podman ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"'
alias container-stats='podman stats'

# Kubernetes shortcuts
alias k='kubectl'
alias pods='kubectl get pods -A'
alias services='kubectl get services -A'

# Resource monitoring
alias resources='btop'
alias network='bandwhich'
alias processes='procs'
```