Horilla HRMS Development Setup Script

This is a comprehensive, reusable bash script for setting up and resetting your Horilla HRMS development environment on Linux. It automates all tedious repetitive steps, allowing you to reset your entire environment (including database) with a single command.[web:1][web:2][web:7]

Features

- **Full Fresh Setup**: Clone repository, create virtual environment, install dependencies
- **Database Reset**: Drop and recreate database schema without re-cloning or reinstalling
- **Fast Teardown**: Clean up environment while keeping repository for quick restarts
- **Skip Options**: Selectively skip steps (clone, database setup) for faster iteration
- **Configuration**: Uses official Horilla environment variables and Django commands
- **Idempotent**: Can be run multiple times safely

Installation

Step 1: Create the Main Setup Script

Create a file named `setup_horilla.sh` in your development directory:

#!/bin/bash

Horilla HRMS Development Setup Script

Usage: ./setup_horilla.sh [options]

Options:

--skip-clone Skip cloning repository (use existing)

--skip-db Skip database setup (keep existing)

--reset-only Only reset database (skip venv, deps)

--help Show this help message

set -e # Exit on error

Color codes for output

RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
NC='\033[0m' # No Color

Configuration

PROJECT_NAME="horilla"
REPO_URL="https://github.com/horilla-opensource/horilla.git"
REPO_BRANCH="1.0"
VENV_NAME="horillavenv"
DB_NAME="horilla_db"
DB_USER="horilla_user"
DB_PASSWORD="${DB_PASSWORD:-horilla123}" # Override with environment variable

Flags

SKIP_CLONE=false
SKIP_DB=false
RESET_ONLY=false

Parse arguments

while [[$# -gt 0]]; do
case $1 in
--skip-clone)
SKIP_CLONE=true
shift
;;
--skip-db)

```
SKIP_DB=true
shift
;;
--reset-only)
RESET_ONLY=true
shift
;;
--help)
grep "^# Usage:" "$0" | sed 's/^# //'
grep "^# " "$0" | sed 's/^# //'
exit 0
;;
*)
echo "Unknown option: $1"
exit 1
;;
esac
done
```

Helper functions

```
print_header() {
echo -e "\n$BLUE═══════════════════════════════════════════════════{NC}"
echo -e "${BLUE}1{NC}"
echo -e "$BLUE═══════════════════════════════════════════════════{NC}\n"
}

print_step() {
echo -e "${GREEN}✓ 1{NC}"
}

print_warning() {
echo -e "${YELLOW}⚠ 1{NC}"
}

print_error() {
echo -e "${RED}✗ 1{NC}"
}

check_command() {
if ! command -v $1 &> /dev/null; then
print_error "$1 is not installed. Please install it first."
exit 1
fi
}
```

Verify prerequisites

verify_prerequisites() {
print_header "Verifying Prerequisites"

```
check_command "python3"
print_step "Python 3 found: $(python3 --version)"

check_command "git"
print_step "Git found: $(git --version)"

check_command "pip"
print_step "Pip found: $(pip3 --version)"
```

}

Clone or update repository

setup_repository() {
print_header "Setting Up Repository"

```
if [ "$SKIP_CLONE" = true ]; then
    if [ ! -d "$PROJECT_NAME" ]; then
        print_error "Repository directory '$PROJECT_NAME' not found. Cannot ski
        exit 1
    fi
    print_warning "Skipping clone (using existing repository)"
    cd "$PROJECT_NAME"
else
    if [ -d "$PROJECT_NAME" ]; then
        print_warning "Repository directory exists. Removing old copy..."
        rm -rf "$PROJECT_NAME"
    fi

    print_step "Cloning Horilla repository..."
    git clone --branch "$REPO_BRANCH" "$REPO_URL" "$PROJECT_NAME"
    print_step "Repository cloned successfully"
    cd "$PROJECT_NAME"
```

```
    fi

    print_step "Working directory: $(pwd)"
```

}

Setup Python virtual environment

setup_venv() {
print_header "Setting Up Virtual Environment"

```
    if [ "$RESET_ONLY" = true ] && [ -d "$VENV_NAME" ]; then
        print_warning "Skipping venv creation in reset mode"
        source "$VENV_NAME/bin/activate"
        return
    fi

    if [ -d "$VENV_NAME" ]; then
        print_warning "Virtual environment exists. Removing old environment..."
        rm -rf "$VENV_NAME"
    fi

    print_step "Creating virtual environment..."
    python3 -m venv "$VENV_NAME"
    print_step "Virtual environment created"

    print_step "Activating virtual environment..."
    source "$VENV_NAME/bin/activate"

    print_step "Upgrading pip..."
    pip install --upgrade pip setuptools wheel
```

}

Install dependencies

install_dependencies() {
print_header "Installing Dependencies"

```
if [ ! -f "requirements.txt" ]; then
    print_error "requirements.txt not found in $(pwd)"
    exit 1
fi

print_step "Installing Python packages from requirements.txt..."
pip install -r requirements.txt
print_step "Dependencies installed successfully"
```

}

Setup environment variables

setup_env_file() {
print_header "Setting Up Environment Variables"

```
if [ -f ".env" ]; then
    print_warning "Existing .env file found. Backing up to .env.backup..."
    cp .env .env.backup
fi

# Create .env file with recommended settings for development
cat > .env << 'ENVEOF'
```

Horilla Development Environment Variables

DEBUG=True
ALLOWED_HOSTS=localhost,127.0.0.1

Database Configuration (PostgreSQL recommended)

DB_ENGINE=django.db.backends.postgresql
DB_NAME=horilla_db
DB_USER=horilla_user
DB_PASSWORD=horilla123
DB_HOST=localhost
DB_PORT=5432

SQLite Alternative (uncomment to use SQLite instead)

DB_ENGINE=django.db.backends.sqlite3

DB_NAME=db.sqlite3

Security (Development Only - Change for Production)

SECRET_KEY=your-secret-key-change-this-in-production
SECURE_SSL_REDIRECT=False
SESSION_COOKIE_SECURE=False

Email (Development - uses console backend)

EMAIL_BACKEND=django.core.mail.backends.console.EmailBackend

ENVEOF

```
print_step ".env file created. Please review and update credentials if needed."
print_warning "Location: $(pwd)/.env"
```

}

Setup PostgreSQL database (optional but recommended)

setup_database() {
print_header "Setting Up PostgreSQL Database"

```
if [ "$SKIP_DB" = true ]; then
    print_warning "Skipping database setup (using existing)"
    return
fi

# Check if PostgreSQL is installed
if ! command -v psql &> /dev/null; then
    print_warning "PostgreSQL client not found. Skipping database setup."
    print_warning "To install: sudo apt install postgresql postgresql-contrib"
    return
fi

print_step "Checking PostgreSQL connection..."

# Try to connect and setup database
if psql -h localhost -U postgres -c "\q" 2>/dev/null; then
    print_step "PostgreSQL is running"

    # Drop existing database if it exists
    print_step "Preparing database..."
    psql -h localhost -U postgres << SQLEOF
```

DROP DATABASE IF EXISTS "$DB_NAME"; $DROPUSERIFEXISTS"DB_USER";
CREATE USER "$DB_USER"$WITHPASSWORD'DB_PASSWORD';
CREATE DATABASE "$DB_NAME"$OWNER"DB_USER";
GRANT ALL PRIVILEGES ON DATABASE "$DB_NAME"$TO"DB_USER";
\q
SQLEOF

```
    print_step "PostgreSQL database created successfully"
else
    print_warning "PostgreSQL is not running or not accessible"
```

```
    print_warning "Starting PostgreSQL service..."
    sudo systemctl start postgresql || true
    sudo service postgresql start || true
    print_warning "Attempting database setup again..."

    # Retry
    psql -h localhost -U postgres << SQLEOF
```

DROP DATABASE IF EXISTS "$DB_NAME"; $DROPUSERIFEXISTS"DB_USER";
CREATE USER "$DB_USER"$WITHPASSWORD'DB_PASSWORD';
CREATE DATABASE "DB_NAMEOWNER"DB_USER";
GRANT ALL PRIVILEGES ON DATABASE "DB_NAMETO"DB_USER";
\q
SQLEOF

```
    print_step "PostgreSQL database created"
fi
```

}

Run Django migrations

run_migrations() {
print_header "Running Django Migrations"

```
print_step "Creating migrations..."
python manage.py makemigrations

print_step "Applying migrations to database..."
python manage.py migrate

print_step "Migrations completed successfully"
```

}

Load demo data (optional)

load_demo_data() {
print_header "Loading Demo Data"

```
# Check if demo data script exists
if [ -f "manage.py" ]; then
    print_warning "Demo data loading is optional. Skipping by default."
    print_warning "To load demo data manually, check official documentation."
fi
```

}

Compile messages for translations

compile_messages() {
print_header "Compiling Messages"

```
print_step "Compiling translation messages..."
python manage.py compilemessages || print_warning "compilemessages failed
```

}

Create superuser

create_superuser() {
print_header "Creating Admin User"

```
print_step "Running Horilla user creation..."
python manage.py createhorillauser

print_step "Admin user created successfully"
```

}

Summary and next steps

print_summary() {
print_header "Setup Complete!"

```
echo -e "${GREEN}Your Horilla HRMS development environment is ready!${NC

echo "Next steps:"
echo "1. Navigate to project directory: cd $PROJECT_NAME"
echo "2. Activate virtual environment: source $VENV_NAME/bin/activate"
echo "3. Start development server: python manage.py runserver"
echo "4. Open browser: http://localhost:8000"
echo ""
echo "Quick commands for future use:"
echo "  • Reset database only: ./setup_horilla.sh --reset-only"
echo "  • Skip clone (faster): ./setup_horilla.sh --skip-clone"
echo "  • Full fresh setup: ./setup_horilla.sh"
echo ""
```

}

Main execution flow

main() {
print_header "Horilla HRMS Development Setup"

```
if [ "$RESET_ONLY" = false ]; then
    verify_prerequisites
    setup_repository
    setup_venv
    install_dependencies
else
    print_step "Reset mode: Skipping prerequisites and venv"
    if [ ! -d "$PROJECT_NAME" ]; then
        print_error "Cannot find project directory. Run full setup first."
        exit 1
    fi
    cd "$PROJECT_NAME"
```

```
    source "$VENV_NAME/bin/activate"
fi

setup_env_file

if [ "$SKIP_DB" = false ]; then
    setup_database
    run_migrations
fi

compile_messages || true

if [ "$RESET_ONLY" = false ]; then
    create_superuser
fi

print_summary
```

}

Run main function

main

Step 2: Make the Script Executable

chmod +x setup_horilla.sh

Usage

Full Fresh Setup (First Time)

./setup_horilla.sh

This will:

- Clone the official Horilla repository (1.0 branch)
- Create Python virtual environment
- Install all dependencies from `requirements.txt`
- Setup PostgreSQL database
- Run Django migrations
- Create admin user

Reset Database Only (After Code Changes)

./setup_horilla.sh --reset-only

This will:

- Drop and recreate the PostgreSQL database
- Run fresh migrations
- **Skip** venv recreation and dependency reinstall
- Takes ~10-15 seconds instead of 2-3 minutes

Skip Repository Clone (Keep Existing)

./setup_horilla.sh --skip-clone

Useful when you already have the repository and just need to reset the environment.

Skip Database Setup

./setup_horilla.sh --skip-db

Skips PostgreSQL setup, useful if you prefer SQLite or database already exists.

Prerequisites

Ubuntu/Debian

sudo apt update
sudo apt install -y python3 python3-pip python3-venv git postgresql postgresql-contrib

Fedora/RHEL

sudo dnf install -y python3 python3-pip git postgresql postgresql-server postgresql-contrib
sudo postgresql-setup --initdb
sudo systemctl start postgresql

Environment Variables

The script creates a `.env` file with development defaults:[web:2][web:12]

DEBUG=True
DB_ENGINE=django.db.backends.postgresql
DB_NAME=horilla_db
DB_USER=horilla_user
DB_PASSWORD=horilla123
DB_HOST=localhost
DB_PORT=5432

To use custom database credentials, set before running:
export DB_PASSWORD="your_custom_password"
./setup_horilla.sh

Database Options

PostgreSQL (Recommended)[web:2]

Already configured in script. Provides better concurrency and scalability.

SQLite Alternative

Edit `.env` file to use:
DB_ENGINE=django.db.backends.sqlite3
DB_NAME=db.sqlite3

Then run:
./setup_horilla.sh --skip-db

Official References

The script follows official Horilla documentation:[web:1][web:7]

1. **Clone Repository**: Uses official GitHub URL and 1.0 branch
2. **Virtual Environment**: Standard Python venv as recommended
3. **Dependencies**: Installs from official `requirements.txt`
4. **Migrations**: Runs `makemigrations` and `migrate` as specified
5. **Admin User**: Uses official `createhorillauser` command
6. **Environment**: Follows official configuration patterns

Troubleshooting

PostgreSQL Connection Error

Start PostgreSQL service

sudo systemctl start postgresql

Or for older systems

sudo service postgresql start

Check status

sudo systemctl status postgresql

Virtual Environment Issues

Manually clean and recreate

rm -rf horillavenv
python3 -m venv horillavenv
source horillavenv/bin/activate
pip install --upgrade pip
pip install -r requirements.txt

Permission Denied

chmod +x setup_horilla.sh

Database User Already Exists

The script automatically drops existing database/user. If it fails:

Manual cleanup (as postgres user)

sudo -u postgres psql
DROP DATABASE IF EXISTS horilla_db;
DROP USER IF EXISTS horilla_user;
\q

Advanced Usage

Using Different Database Password

DB_PASSWORD="my_secure_pass" ./setup_horilla.sh

Only Run Migrations (No Database Reset)

cd horilla
source horillavenv/bin/activate
python manage.py migrate

Access PostgreSQL Directly

psql -U horilla_user -d horilla_db -h localhost

What Gets Reset With `--reset-only`

✓ Database schema (all tables dropped and recreated)
✓ Django migrations
✓ Demo data (if any)

✗ Virtual environment (preserved for speed)
✗ Installed packages (preserved for speed)
✗ Repository code (preserved)
✗ Static files (preserved)

Next Steps

1. **Copy the script**: Save `setup_horilla.sh` to your development directory
2. **Make it executable**: `chmod +x setup_horilla.sh`
3. **Run first setup**: `./setup_horilla.sh`
4. **Start developing**: `cd horilla && source horillavenv/bin/activate && python manage.py runserver`
5. **When you need fresh database**: Just run `./setup_horilla.sh --reset-only`

Related Documentation

- [Horilla Official Installation][web:7]
- [Horilla Database Configuration][web:2]
- [Django Migrations Documentation][web:13]
- [Environment Variables Guide][web:12]