Tutorial: Creating and Running Conda Environments Using Shell Scripts

This tutorial provides a step-by-step guide for creating reproducible Conda environments and standardized run scripts using shell (`.sh`) files. The goal is to ensure that teams follow a consistent, reliable, and repeatable process when setting up and running environments for projects.

# 1. Prerequisites

- Conda (Anaconda or Miniconda) must be installed.  
- Verify installation using: `conda --version`  
- Ensure required Python version is supported by Conda.  
- Stable internet connection is needed for package installation.  
- Sufficient permissions to create environments and install dependencies.  
- A `requirements.txt` file (optional but recommended) listing dependencies.  
- On Windows: Run the script using Git Bash or WSL.

# 2. Dependencies & Tools

The scripts can be customized to install the following:  
- Required Python version.  
- Project-specific dependencies listed in `requirements.txt`.  
- Jupyter Notebook and ipykernel (if environment is to be used in Jupyter).  
- Any additional libraries specific to the project.

# 3. Environment Naming Conventions

- Use clear and descriptive names (e.g., `<project\_name>\_env`).  
- Avoid spaces or special characters.  
- Ensure consistency across the team.

# 4. Shell Script Best Practices

- Always start with a shebang (`#!/bin/bash`).  
- Use variables for environment name and Python version.  
- Add echo logs for better readability.  
- Include error handling after critical steps.  
- Use `export` to set environment variables where needed.  
- Make the script executable before running: `chmod +x <script\_name>.sh`

# 5. Common Pitfalls to Avoid

- Forgetting to `source` Conda’s profile script before activating.  
- Typos in environment name.  
- Missing `requirements.txt` file.  
- Running the script from within another Conda environment.  
- Using an unsupported Python version in Conda.

# 6. Generalized Environment Creation Script (create\_env.sh)

#!/bin/bash  
  
# Variables (replace <> with actual values)  
ENV\_NAME="<environment\_name>"  
CONDA\_PYTHON\_VERSION="<python\_version>"  
  
echo ">>> Creating Conda environment: $ENV\_NAME with Python $CONDA\_PYTHON\_VERSION"  
  
# Step 1: Create environment  
conda create -y -n $ENV\_NAME python=$CONDA\_PYTHON\_VERSION  
if [ $? -ne 0 ]; then  
 echo "❌ Error: Failed to create Conda environment."  
 exit 1  
fi  
  
# Step 2: Activate environment  
echo ">>> Activating Conda environment..."  
source "$(conda info --base)/etc/profile.d/conda.sh"  
conda activate $ENV\_NAME  
  
# Step 3: Install dependencies (if requirements.txt exists)  
if [ -f "requirements.txt" ]; then  
 echo ">>> Installing dependencies from requirements.txt..."  
 pip install -r requirements.txt  
else  
 echo "⚠️ No requirements.txt found. Skipping pip installs."  
fi  
  
# Optional: Install Jupyter support  
conda install -y notebook ipykernel  
python -m ipykernel install --user --name $ENV\_NAME --display-name "$ENV\_NAME"  
  
echo "✅ Environment $ENV\_NAME created successfully!"

# 7. Verification After Environment Creation

- Verify environment creation:  
 `conda env list`  
- Check Python version:  
 `python --version`  
- If Jupyter was installed, confirm kernel availability in Notebook.

# 8. Generalized Run Script (run.sh)

A `run.sh` file helps standardize how projects are started after setting up the Conda environment. This avoids confusion when multiple team members run services in different ways.

## Best Practices

- Always activate the correct Conda environment before running the service.  
- Use `echo` logs for better visibility of what the script is doing.  
- Use variables for environment name, entry point, and app type for flexibility.  
- Add error handling for unknown or unsupported commands.  
- Use `export` for setting environment variables like `FLASK\_APP`, `APP\_ENV`, etc.

## Example Template: run.sh

#!/bin/bash  
  
# Variables (replace <> as needed)  
ENV\_NAME="<environment\_name>"  
APP\_ENTRY="<entry\_point\_file\_or\_module>"  
APP\_TYPE="<uvicorn|streamlit|flask|jupyter|jupyterlab|custom>"  
  
# Optional environment variables  
export APP\_ENV="development"  
export PYTHONUNBUFFERED=1  
  
echo ">>> Activating Conda environment: $ENV\_NAME"  
source "$(conda info --base)/etc/profile.d/conda.sh"  
conda activate $ENV\_NAME  
  
# Decide which service to run  
case "$APP\_TYPE" in  
 "uvicorn")  
 echo ">>> Starting Uvicorn server..."  
 python -m uvicorn $APP\_ENTRY --host 0.0.0.0 --port 8000 --reload  
 ;;  
 "streamlit")  
 echo ">>> Starting Streamlit app..."  
 streamlit run $APP\_ENTRY  
 ;;  
 "flask")  
 echo ">>> Starting Flask app..."  
 export FLASK\_APP=$APP\_ENTRY  
 flask run --host=0.0.0.0 --port=5000  
 ;;  
 "jupyter")  
 echo ">>> Starting Jupyter Notebook..."  
 jupyter notebook  
 ;;  
 "jupyterlab")  
 echo ">>> Starting Jupyter Lab..."  
 jupyter lab  
 ;;  
 "custom")  
 echo ">>> Running custom command..."  
 python $APP\_ENTRY  
 ;;  
 \*)  
 echo "❌ Error: Unknown APP\_TYPE '$APP\_TYPE'"  
 echo "Valid options: uvicorn, streamlit, flask, jupyter, jupyterlab, custom"  
 exit 1  
 ;;  
esac

## How to Use run.sh

1. Update the variables `<environment\_name>`, `<entry\_point\_file\_or\_module>`, and `<APP\_TYPE>`.  
 Example:  
 ```bash  
 ENV\_NAME="Unpack"  
 APP\_ENTRY="app:app"  
 APP\_TYPE="uvicorn"  
 ```  
  
2. Make the script executable:  
 ```bash  
 chmod +x run.sh  
 ./run.sh  
 ```

## Supported Cases

- \*\*Uvicorn\*\*: `python -m uvicorn app:app --reload`  
- \*\*Streamlit\*\*: `streamlit run app.py`  
- \*\*Flask\*\*: `flask run` (with `FLASK\_APP` set)  
- \*\*Jupyter Notebook\*\*: `jupyter notebook`  
- \*\*Jupyter Lab\*\*: `jupyter lab`  
- \*\*Custom\*\*: Run any Python file/module directly