Visual Studio - Publish to Linux and Restart Service

To stop and restart a Linux service automatically when publishing from Visual Studio to a local folder, you must modify your **Publish Profile (`.pubxml`)** to include post-publish commands. Since Visual Studio runs on Windows, these commands typically trigger a script (like PowerShell or a WinSCP script) to remotely execute the `systemctl` commands on the Linux server.

NOTE: In the exercise our service is called `api.service`

1. Update the Publish Profile

Visual Studio doesn't have a built-in "Post-publish" box in the UI for folder publishing, so you must edit the `.pubxml` file manually.

1. In **Solution Explorer**, expand `Properties` > `PublishProfiles`.
2. Open your `.pubxml` file (e.g., `FolderProfile.pubxml`).
3. Add a custom `Target` at the end of the file, before the closing `</Project>` tag:

```
<Target Name="CustomActionsAfterPublish" AfterTargets="AfterPublish">
  <Exec Command="powershell -ExecutionPolicy Bypass -File
&amp;quot;$(ProjectDir)Scripts\RestartLinuxService.ps1&amp;quot;" />
</Target>
```

2. Generate SSH Key on Your Windows Machine (once)

Open PowerShell as your normal user and run:

```
ssh-keygen -t ed25519 -C "visualstudio-publish-key"
# Press Enter for default location (~/.ssh/id_ed25519)
# Leave passphrase empty (for fully automated) or set one (then use ssh-agent)
```

3. Copy the Public Key to Your Linux Server (as root or your user)

```
# Replace with your actual server
type $HOME\.ssh\id_ed25519.pub | ssh root@linux-server "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys
&& chmod 600 ~/.ssh/authorized_keys && chmod 700 ~/.ssh"
```

Enter the root password **one time** during this setup.

4. Create Your `RestartLinuxService.ps1` Script

Since your publish target is a folder, you likely still need to move the files to Linux and then restart the service. You can use **PowerShell** with SSH to handle this.

Create or replace the script with this clean version (no password needed):

```
# RestartLinuxService.ps1
# Run after Visual Studio publish to stop/restart the systemd service

$server = "linux-server"
$user   = "root"   # or a sudo-enabled user

Write-Host "Stopping api.service on $server..." -ForegroundColor Yellow
ssh -o StrictHostKeyChecking=no -o BatchMode=yes "${user}@${server}" "sudo systemctl stop api.service"

# Optional: If you also want to sync files here (instead of relying only on VS publish folder profile)
# Write-Host "Syncing files..."
# scp -r -o StrictHostKeyChecking=no "C:\Path\To\Your\Publish\Output\*"
"${user}@${server}:/path/to/your/app/"

Write-Host "Restarting api.service on $server..." -ForegroundColor Green
ssh -o StrictHostKeyChecking=no -o BatchMode=yes "${user}@${server}" "sudo systemctl restart api.service"

# Verify status (optional)
ssh -o StrictHostKeyChecking=no -o BatchMode=yes "${user}@${server}" "sudo systemctl status api.service --
no-pager -l"
```

Notes:

- -o BatchMode=yes fails fast if something is wrong (no interactive prompts).
- -o StrictHostKeyChecking=no skips the host key prompt on first run (you can remove it after the first successful connection).

5. Modifying sudoers

Make sure the Linux user (root or a dedicated deploy user) can run sudo systemctl without a password.

Option 1: On the Linux server, edit sudoers:

```
sudo visudo
```

Add a line like:

```
deployuser ALL=(ALL) NOPASSWD: /usr/bin/systemctl stop api.service, /usr/bin/systemctl restart api.service, /usr/bin/systemctl status api.service
```

Option 2: Add to the `/etc/sudoers.d` directory

Using /etc/sudoers.d is the recommended and best practice way to add this kind of limited sudo permission. It keeps your changes clean, separate from the main /etc/sudoers file, and easier to manage or remove later.

Create a Proper sudoers.d Entry

On your Linux server `linux-server`, run these commands as root:

1. Create the file safely using `visudo` (this checks syntax automatically)

```
sudo visudo -f /etc/sudoers.d/deploy-api
```

2. Paste the following content into the editor that opens:

```
# Allow the deployuser to manage the api.service without entering a password
# This is very limited - only stop, restart, and status for this specific service

deployuser ALL=(ALL) NOPASSWD: /usr/bin/systemctl stop api.service, \
                               /usr/bin/systemctl restart api.service, \
                               /usr/bin/systemctl status api.service
```

Important notes on the rule:

- Use the **full path** /usr/bin/systemctl (most common on modern Ubuntu/Debian/RHEL).
- You can add more commands if needed (e.g., start, reload, enable).
- The backslashes (\) allow the rule to span multiple lines for readability.
- `deployuser` should be replaced with the actual username you use in your SSH script (e.g., deploy or apiuser — **do not use root** for this).

Save and exit the editor (Ctrl+O, Enter, Ctrl+X in nano).

Verify the Rule Works

After saving, test it from the deployuser account (or via SSH):

```
# Switch to the deploy user if needed
su - deployuser

# Test the allowed commands (should NOT ask for password)
sudo systemctl status api.service
sudo systemctl stop api.service
sudo systemctl restart api.service
```

You can also check what the user is allowed to run:

```
sudo -l -U deployuser
```

Updated PowerShell Script (Using the Non-Root Deploy User)

Now update your `RestartLinuxService.ps1` to use the dedicated deploy user instead of root:

```
# RestartLinuxService.ps1 - Improved version

$server = "linux-server"
$user   = "deployuser"   # <-- Change to your non-root deploy user

Write-Host "Connecting to $server as $user..." -ForegroundColor Cyan

# Stop the service
Write-Host "Stopping api.service..." -ForegroundColor Yellow
ssh -o StrictHostKeyChecking=no -o BatchMode=yes "${user}@${server}" "sudo systemctl stop api.service"

# Restart the service
Write-Host "Restarting api.service..." -ForegroundColor Green
ssh -o StrictHostKeyChecking=no -o BatchMode=yes "${user}@${server}" "sudo systemctl restart api.service"

# Optional: Show brief status
Write-Host "Service status:" -ForegroundColor Cyan
ssh -o StrictHostKeyChecking=no -o BatchMode=yes "${user}@${server}" "sudo systemctl status api.service --no-pager -l" | Select-Object -First 15
```

Extra Security Tips

- **File permissions** — The file /etc/sudoers.d/deploy-api must be 0440 (readable only by root):

```
sudo chmod 0440 /etc/sudoers.d/deploy-api
sudo chown root:root /etc/sudoers.d/deploy-api
```

- **Filename convention** — Many admins prefix with a number (e.g., 10-deploy-api or 99-deploy-api) to control loading order.
- **Least privilege** — This rule is already quite tight. Avoid adding ALL or wildcard * unless absolutely necessary.
- If you want even tighter control, you can define a `Cmnd_Alias` first in the same file, see below for the link

Finally

In both options, your existing `.pubxml` target stays exactly the same as in step 1

```
<Target Name="CustomActionsAfterPublish" AfterTargets="AfterPublish">
  <Exec Command="powershell -ExecutionPolicy Bypass -File
&amp;quot;$(ProjectDir)Scripts\RestartLinuxService.ps1&amp;quot;" />
</Target>
```

6. Alternative: If You Must Use Password (Not Recommended)

If key auth is blocked (e.g., company policy), you can use **sshpass** (install via Chocolatey: choco install sshpass) or a simple PowerShell wrapper.

Using sshpass (install once):

```
# In RestartLinuxService.ps1
$password = "YourSuperSecretPassword"   # <-- WARNING: Plain text! Very insecure

$server = "linux-server"
$user   = "root"

sshpass -p $password ssh -o StrictHostKeyChecking=no "${user}@${server}" "sudo systemctl restart
api.service"
```

Even worse for security: store the password encrypted with `ConvertFrom-SecureString` and load it at runtime, but it's still risky for automated builds.

7. Bonus Tips

- **Run as non-root** — Create a dedicated deploy user on Linux with limited sudo rights.
- **File sync** — The publish profile in Visual Studio can already target a network share or use **Web Deploy** / **Folder publish with rsync/scp** in the profile. Then your script only needs to handle stop → restart.
- **Error handling** — Add try/catch and check $LASTEXITCODE after each ssh call.
- **Test first** — Run the .ps1 manually from PowerShell before relying on the publish step.

8. `Cmnd_Alias` setup for ease of multi-commands

Click Here

Revision #6
Created 6 April 2026 14:48:06 by Steve Ling
Updated 6 April 2026 15:39:52 by Steve Ling