

Here is a **complete and detailed consolidated writeup** for delivering and maintaining your .NET Core + Dapr-based microservices application in Windows environments (non-containerized), supporting both **initial installation** and **periodic updates**, with and without installer packages.

Delivery Strategy for .NET + Dapr Microservices on Windows (Non-Docker)

This guide explains how to:

- Deliver your distributed .NET microservices application on Windows machines.
- Run each microservice with Dapr sidecar.
- Support multi-server and multi-client deployment.
- Provide both ZIP-based and installer-based (NSIS) installation options.
- Manage upgrades via versioned update packages.

Assumptions

- Your application has multiple microservices (e.g., OrderService, InventoryService).
- Each microservice is a self-contained .NET Core app with its appsettings.json.
- You're using [Dapr](#) with components like pub/sub, Redis state store, etc.
- Services are deployed as Windows Services (optional but recommended).
- You support periodic updates and need easy rollout.

Preparatory Steps (Development and Packaging)

1. Organize Your Service Output

Each service must be published using:

```
dotnet publish -c Release -r win-x64 --self-contained true /p:PublishSingleFile=true
```

Folder structure:

```
\build
\OrderService\
  OrderService.exe
  appsettings.json
\InventoryService\
  InventoryService.exe
  appsettings.json
```

2. Define dapr and components

Include:

- dapr.exe (or have installer script fetch it)
- Your components/ YAML files for Redis, pub/sub, etc.
- env-config.json (holds environment-wide settings)

3. Version Tracking

Include:

- version.json: Lists app version and individual service versions.
- changelog.txt: Tracks what's changed across builds.

Example:

```
{
  "appVersion": "1.0.2",
  "services": {
    "OrderService": "1.0.2",
    "InventoryService": "1.0.1"
  }
}
```

Delivery Package Formats

Option A: ZIP-Based Delivery

Two packages:

A1. Initial Setup ZIP

- MyApp-Windows-Package.zip
- Contains:
 - All service folders with executables
 - Dapr + Redis
 - components/
 - PowerShell Scripts:
 - install-all.ps1 – installs Dapr, sets up folders
 - register-service.ps1 – registers each microservice as Windows service
 - run-with-dapr.ps1 – optionally starts service manually with Dapr
 - uninstall-all.ps1 – stops and removes all services
 - README.md

A2. Update ZIP

- MyApp-Update-Package.zip
- Contains:
 - Only changed service folders
 - version.json, changelog.txt
 - update.ps1 – stops the Windows service, replaces .exe, restarts

Option B: NSIS Installer

For user-friendly initial setup:

- MyAppInstaller.exe (built with [NSIS](#))
- Features:
 - Installs binaries to C:\Program Files\MyApp
 - Optionally sets environment variables

- Adds Windows services
- Auto-starts Dapr sidecars
- Writes install path and version to registry

Steps for Deployment

Initial Setup (ZIP or NSIS)

Using ZIP

1. Extract MyApp-Windows-Package.zip to C:\MyApp
2. Open PowerShell (Admin)
3. Run:

Set-ExecutionPolicy Bypass -Scope Process -Force

.\scripts\install-all.ps1

.\scripts\register-service.ps1

Optionally modify .env or appsettings.json before running.

Using NSIS Installer

1. Run MyAppInstaller.exe as Administrator
2. Choose install location (default: C:\Program Files\MyApp)
3. Installer does:
 - a. Copy files
 - b. Install Dapr and Redis (if needed)
 - c. Register services
 - d. Start services with Dapr

Periodic Update

1. Extract MyApp-Update-Package.zip
2. Open PowerShell (Admin)
3. Run:

```
.\update.ps1 -BaseDir "C:\MyApp"
```

This script:

- Parses version.json
- Stops affected services
- Replaces executables
- Restarts them

Service Registration (PowerShell)

Each service is registered like:

```
New-Service -Name "OrderService" `
-BinaryPathName "C:\MyApp\OrderService\OrderService.exe" `
-DisplayName "Order Service" -StartupType Automatic
```

Dapr is started in sidecar mode for each service:

```
dapr run --app-id orderservice --app-port 5001 --dapr-http-port 3501 --components-
path ./components -- dotnet OrderService.exe
```

Use run-with-dapr.ps1 to run manually if needed.

Supporting Multiple Clients & Multi-Server Environments

- Use env-config.json to define per-client/env parameters (ports, URLs, keys).
- Use components/ folder for client-specific Dapr config (state store, bindings, etc).
- Automate deployment across servers via a central orchestration script or Ansible.

Security Considerations

- Use HTTPS with reverse proxy (e.g., IIS, nginx).
- Encrypt sensitive config values (e.g., connection strings).
- Use Windows Service permissions to restrict service access.
- Configure Dapr access control policies (ACL) for APIs and actors.

Automation Recommendations

- Use a build pipeline (e.g., Azure DevOps, GitHub Actions) to:
 - Generate ZIPs and NSIS installers
 - Update version.json dynamically
 - Tag releases

Summary

Task	ZIP Package	NSIS Installer
Initial Setup	Manual unzip + script	One-click installer
Windows Service	Included in scripts	Auto-registered
Updates	update.ps1 script	Can be done manually
Multi-client Config	Yes via env-config.json	Yes
Dapr Integration	Full support	Full support