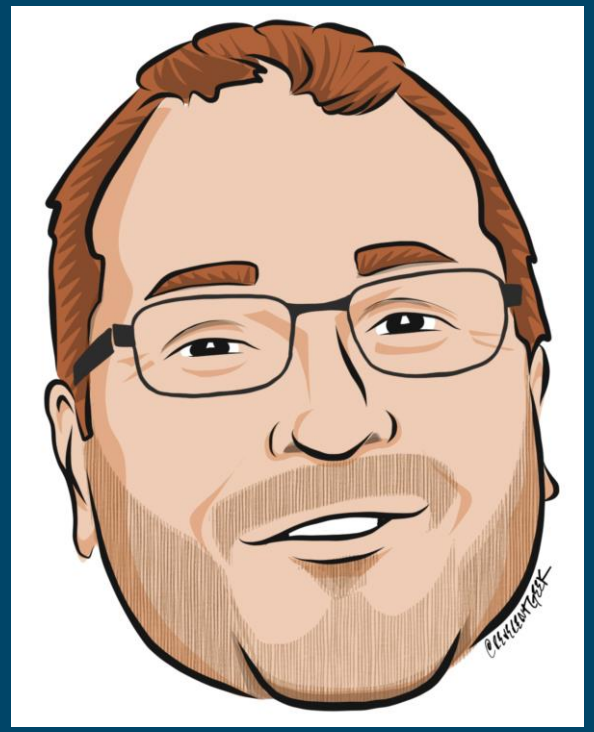
Automating the Software Deployment Lifecycle

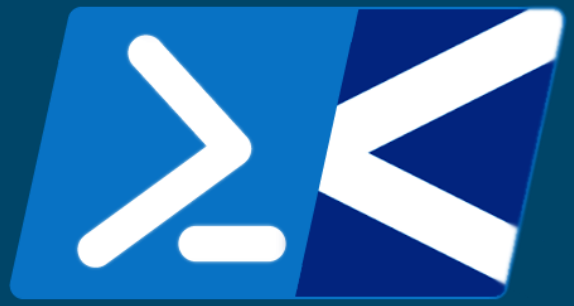
with Chocolatey, Jenkins and PowerShell

Paul Broadwith
@pauby
https://blog.pauby.com

Who Am I?

- Paul Broadwith, Glasgow, Scotland
- 25+ years in defence, government, financial sectors
- Lead Engineer on Boxstarter and Chocolatey cChoco DSC Resource





Agenda

- What is Chocolatey?
- Chocolatey Sources;
- Internalizing packages;
- Recommended Organizational Architecture;
- Common scenarios where Chocolatey automation will help you;

- Based on this blog post: https://blog.pauby.com/post/getting-started-with-chocolatey-and-jenkins/

Before We Start!

What Is Chocolatey?

A Definition Of Chocolatey

> Chocolatey is a package manager for Windows, like apt-get or yum but for Windows. It was designed to be a decentralized framework for quickly installing applications and tools that you need. It is built on the NuGet infrastructure currently using PowerShell as its focus for delivering packages from the distros to your door, err computer.

Chocolatey manages Packages

Packages manage Installers

Chocolatey Package Sources

Where do packages come from?

Chocolatey Sources

- Where packages come from;
- C4B comes with two Chocolatey sources by default:
  - `chocolatey` – Chocolatey Community Repository
  - `chocolatey.licensed` - Chocolatey Community Repository cached binaries;
- Add your own sources:
  - Repository manager: Artifactory, Nexus, ProGet, Azure DevOps Feeds
  - Local folder

Azure DevOps Artifacts Feeds

- Requires a license to use – only 5 provided;
  - Create tokens for multiple 'users';
  - See https://blog.pauby.com/post/chocolatey-repository-using-azure-devops-artifacts-feed/
- Chocolatey considerations:
  - Cannot use `choco push` as it requires to push to a URL and supports only an `--api-key` (Azure DevOps pipeline?);
  - Use `nuget push` instead as it allows you to push to a named source (where the username and password is defined);
  - Turn off repository optimizations: `choco feature disable -n=usePackageRepositoryOptimizations`

Demo 1

Chocolatey Sources.

Internalizing Packages

Keeping it in the family.

Why Internalize Packages?

- What is ‘package internalization’?
- Organizations recommended to disable the default sources:
  - Reliability
  - Trust
  - Bandwidth
  - Copyright Restrictions
- Using the default `chocolatey` source is subject to:
  - rate limiting;
  - excessive download limiting;

C4B Package Internalizer

- Automatically internalizes the vast majority of packages;
- Very fast;
- Don’t reinvent the wheel;
- Automation!

Demo 2

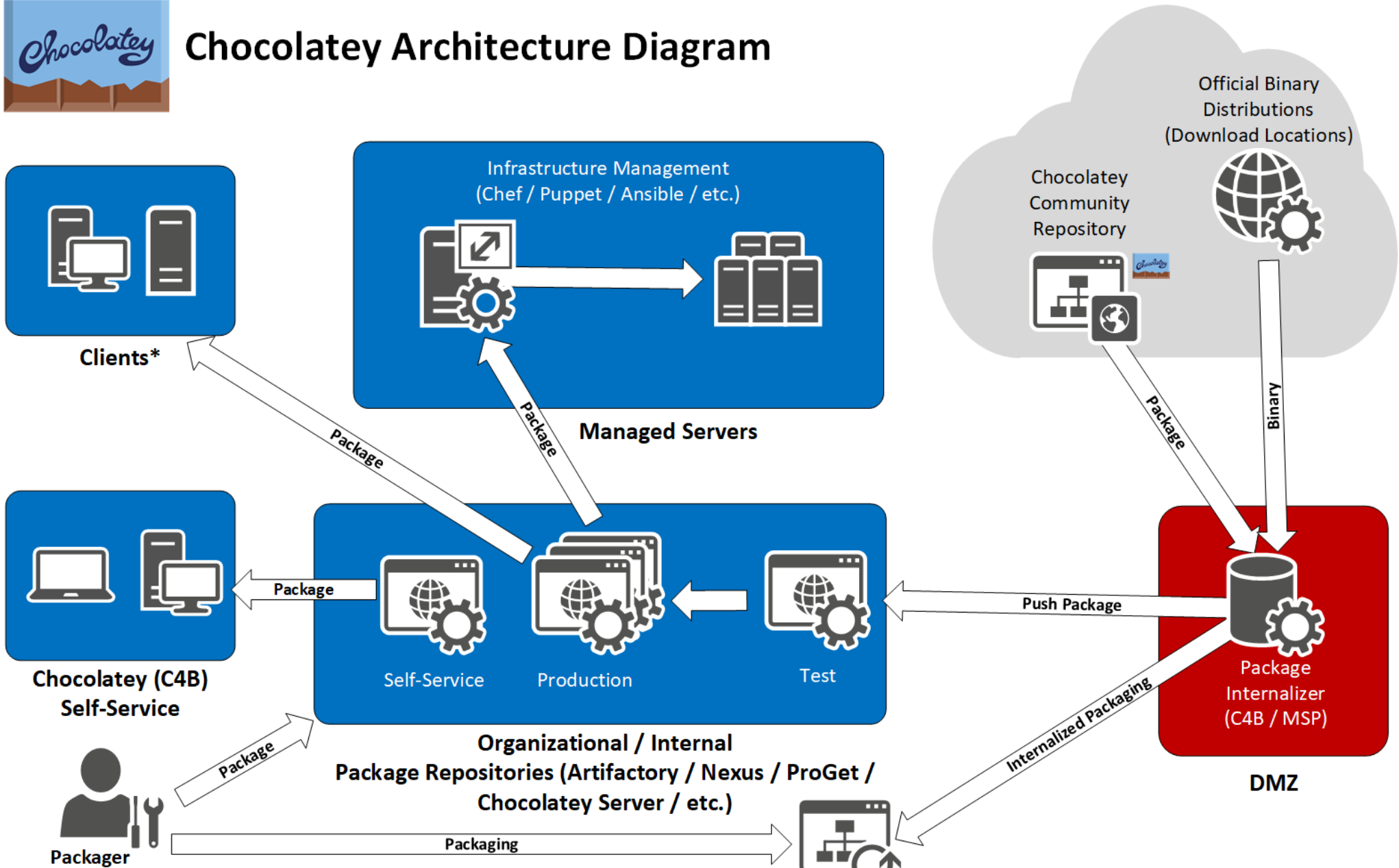
Package Internalization.

Organizational Architecture

Every organisation is a snowflake.

(shield your eyes!)

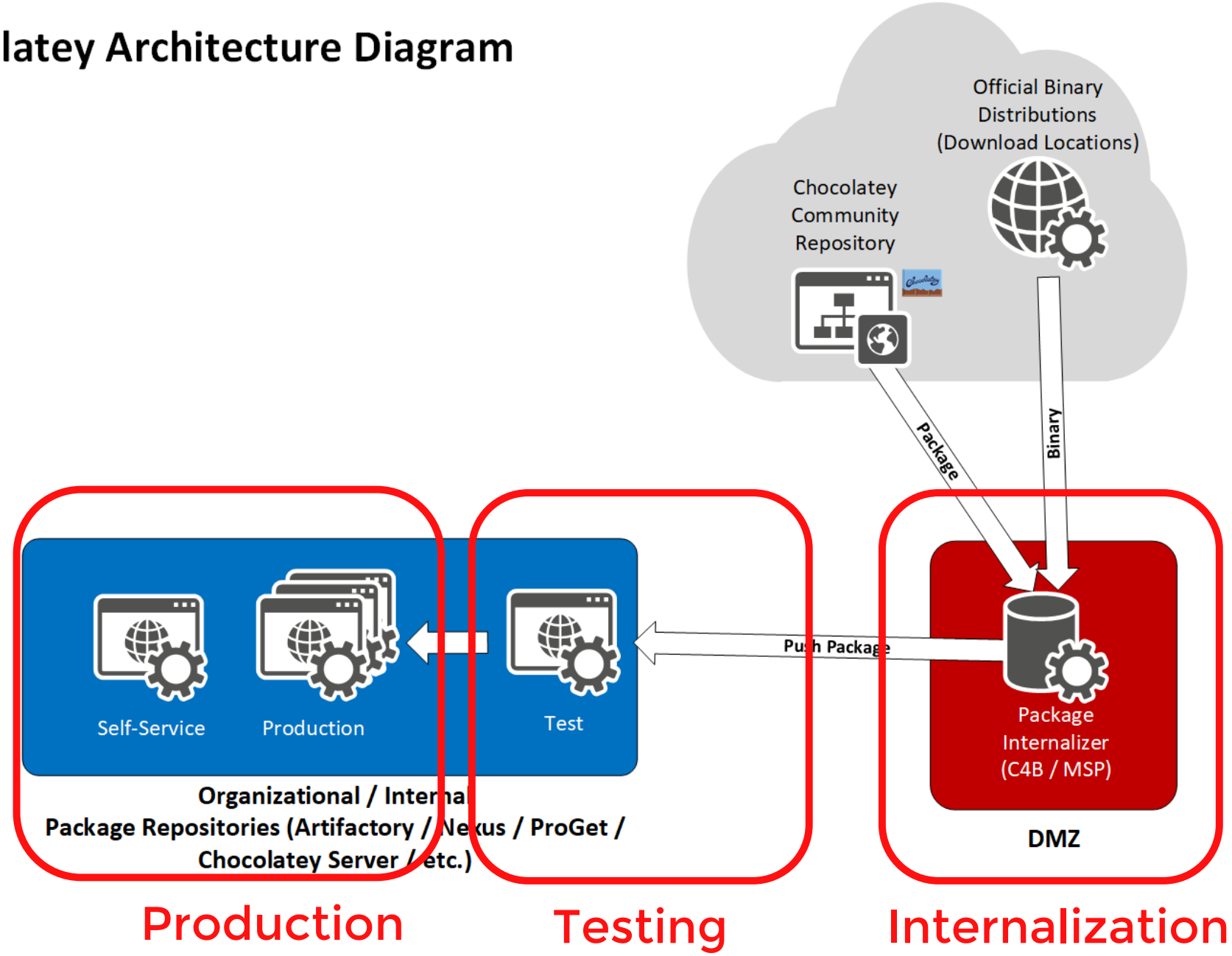
Chocolatey Architecture Diagram



* Clients could be managed directly or through end point management solutions like Microsoft SCCM, Microsoft Intune, Altiris, LanDesk, etc. Also Chocolatey's Central Management.

v1.4 17 July 2018

Chocolatey Architecture Diagram



Tools We Need

Package Repositories – Azure DevOps Artifacts Feeds

Code - PowerShell

Automation - Jenkins

Demo 3

Lets look at the Jenkins jobs.

```powershell
# Jenkins - Internalize Package and Push to Test Repo

 $temp = Join-Path -Path $env:TEMP -ChildPath ([GUID]::NewGuid()).Guid
New-Item -Path $temp -ItemType Directory | Out-Null
Write-Verbose "Created temporary directory '$temp'."

Set-Location (Join-Path -Path $env:SystemDrive -ChildPath 'scripts')
.\Internalize-ChocoPackage.ps1 `
    -Name $env:P_PKG_LIST `
    -OutputDirectory $temp `
    -Verbose

if ($LASTEXITCODE -eq 0) {
    Set-Location $temp
    Get-ChildItem -Path *.nupkg | ForEach-Object {
        nuget push $_.Name -source $env:G_TEST_REPO -apikey $env:G_API_KEY `
            -configfile c:\nuget\nuget.config
    }
}
```

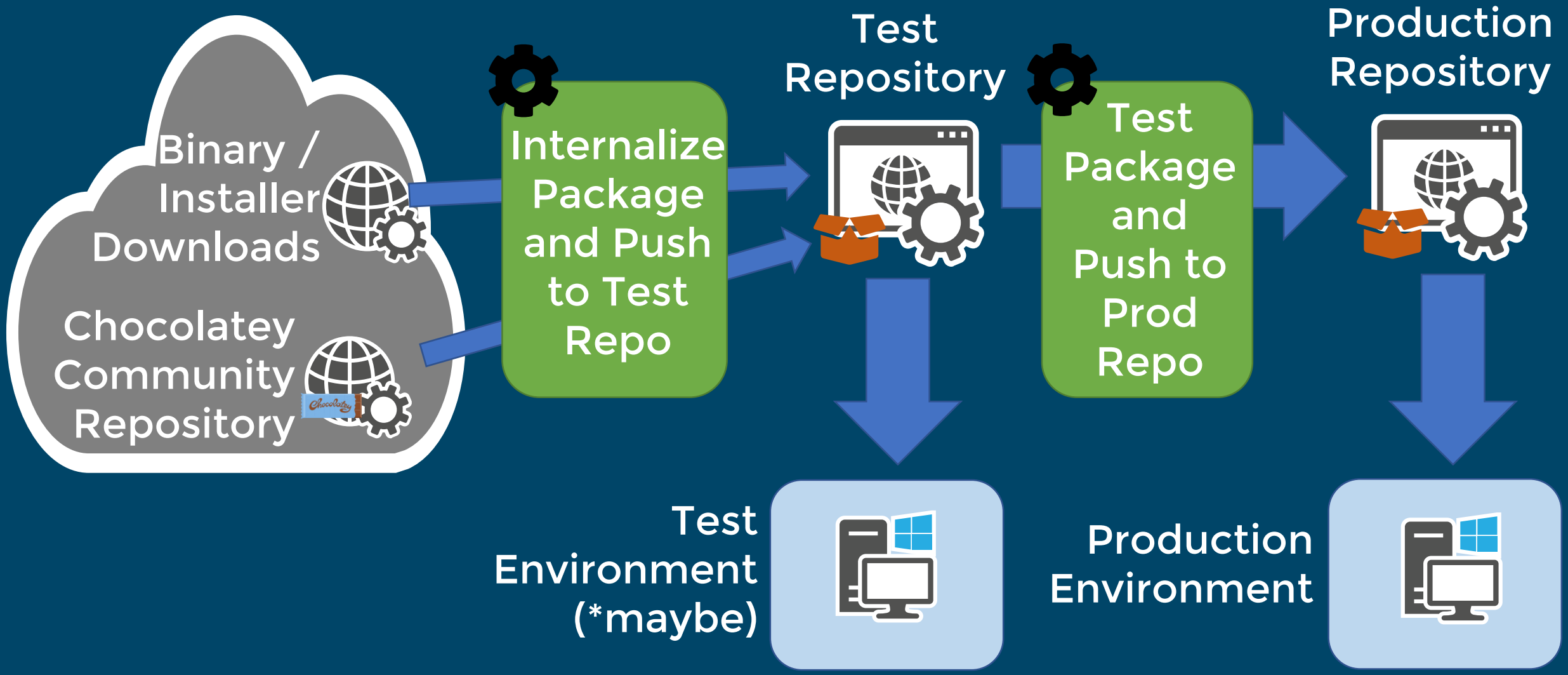
```powershell
# Jenkins - Test Package and Push to Prod Repo

Set-Location (Join-Path -Path $env:SystemDrive -ChildPath 'scripts')
.\Update-ProdRepoFromTest.ps1 `
    -ProdRepo $env:G_PROD_REPO `
    -ProdRepoApiKey $env:G_API_KEY `
    -TestRepo $env:G_TEST_REPO `
    -Verbose
```

```powershell
# Jenkins - Upgrade Outdated Packages in Test Repository

Set-Location (Join-Path -Path $env:SystemDrive -ChildPath 'scripts')
.\Get-UpdatedPackage.ps1 -LocalRepo $env:P_LOCAL_REPO_URL `
    -LocalRepoApiKey $env:P_LOCAL_REPO_API_KEY `
    -RemoteRepo $env:P_REMOTE_REPO_URL `
    -Verbose
```

Jenkins Configuration



Common Scenarios

You helping Chocolatey help you.

Demo 4

Jenkins PowerShell Scripts and Common Scenarios

Suggested Improvements.

Ideas to extend and customise.

Suggested Improvements

- Schedule Updating Packages from Chocolatey Community Repository;
- Write your own Pester Tests that your organization needs;
- When packages are published to production send a message via email / Slack / Teams or whatever you use;

Summary

- What Chocolatey is and how sources work;
- Understand package internalization and why it's important for organizations;
- How to apply automation to Chocolatey package testing and deployment;

Questions?

Paul Broadwith

pauby.com

@pauby

github.com/pauby

linkedin.pauby.com

pau.by/talks