Release Pipelines in Microsoft Ecosystems

Warren Frame, Harvard University

Michael Greene, Microsoft
whoami

• Warren Frame
  • Research Computing at Harvard University
  @pscookiemonster
  Ramblingcookiemonster
  wframe

• Michael Greene
  • Enterprise Cloud Engineering CAT Team at Microsoft
  @migreene
  mgreenegit
  migreene

bit.ly/lisa16pipeline
Stuff

- Slides
- Demos!
- Cleanup,
Configuration as Code

- Everything-as-a-service, APIs galore
- Living documentation
- Abstract out complexity. Scripts -> Modules -> DSC -> key:value
- PowerShell DSC is a platform that all solutions can use to deploy and manage Windows Server
- Azure Resource Manager templates
- You still need to know the underlying systems you will manage
- Release pipelines can bring sanity and consistency to managing this
Release Pipeline

Why?

aka.ms/trpm
Release Pipelines

Prod environment (etc.)...
- Systems / Services
- Modules
- Scripts
- Config files
Example Workflow

Make a change, push to source control*

Build system does the rest. For example:

- Run tests against your code
- Spin up test services/infrastructure for more tests
- Build artifacts (packages, configs, etc.)
- Deploy things (artifacts, systems, services, etc.)

* You might run through source-build-test loops locally until happy, before pushing.
Tooling

“a bunch of random open source projects bound together with duct tape and chewing gum”
Tools: Source

Git? Mercurial? SVN?

CLI:
• Git for Windows
• PoshGit

GUI:
• GitHub Desktop
• Atlassian SourceTree
• Many others
Demo: Source

Git

Visual Studio Code
Tools: Build Systems

- Jenkins, GitLab CI, VSTS, etc.
- Prefer build-as-code
  - e.g. Jenkinsfile, appveyor.yml, .gitlab-ci.yml
Tools: Build Automation

- Invoke-Build, psake
- Similar to rake, make, bake, cake, grunt, gulp, msbuild, etc.

```powershell
$SomeSharedVariable = 'or not'

# Entry points. Dot is default.
Task . Deploy

Task Deploy Init, Build, Test, Deployment

Task Init { "Initialize things..." }

Task Build { "Build a thing... $SomeSharedVariable" }

Task Test Init, Build, { "Assert a thing..." }

Task Deployment { "Deploy a thing..." }
```
Demo: Build

TFS 2017
psake (build automation)
github.com/powershell/demo_ci
Tools: Testing

- **Pester:** Test framework
- **poshspec:** infrastructure testing
- **OVF:** Operation-Validation-Validation-_framework - simplify organizing, execution, and sharing of tests.
Demo: Test

Pester
poshspec
Tools: Release

- **Octopus Deploy and VSTS**
  - Many pre-canned tasks
  - Flexible
  - Pretty
  - Potentially $$

- **PSDeploy**
  - Some pre-canned tasks
  - Deployment as code
  - Poorly written
  - Open source

- **Random PowerShell code**
  - Fun to read and maintain!
Demo: Release

TFS 2017 - Release management
Tools: Test Harness

- Test-Kitchen
- Not just for Chef
- Roughly:
  - Run tests with a **verifier** (Pester)
  - against **platforms** (different vagrant boxes)
  - converged with a **provisioner** (dsc)
  - with the lifecycle managed by a **driver** (vagrant)
  - And test, configuration, other files copied to **platforms** via a **transport** (WinRM)

Drivers
- Amazon EC2
- Azure Resource Manager
- DigitalOcean
- Docker
- Google Compute Engine
- Hyper-V
- OpenStack
- Vagrant
- vRealize Automation, Orchestrator
- vSphere

Provisioners
- Ansible
- CFEngine
- Chef Solo, Zero
- DSC
- Puppet
- Salt
- Shell

Verifiers
- Inspec
- Pester
- Shell (Bats, Serverspec, etc.)
Example Pipeline

Source: GitHub
Build: AppVeyor

- Build dependencies: PSDepend
- Build automation: Invoke-Build
- Build helpers: BuildHelpers

Test: Pester
Release: PSDeploy
Demo: Example Pipeline

https://github.com/RamblingCookieMonster/lisa-kitchen-demo
What about...

• Secrets
  • In-source control
  • Built into build system?
  • Secret management – vault, passwordstate, Secret Server, credstash, etc.

• Images
  • Packer!
  • Images-as-code
  • Build images for Amazon, VirtualBox, Azure, Hyper-V(ish), etc.
Where to start

• Source Control and/or Tests over entire pipeline at once
• Existing tools over resume-driven-development
• New service(s) / value proposition over re-engineering everything
• No luck in house? Play with GitHub+AppVeyor, VSTS, etc.
Next steps

• Open source projects could use your help!
• JIT provisioning or a dynamic pools of Windows build agents
• Windows Docker containers for testing
• Focus on ephemeral deployments over incremental changes
• Plan for day 100
References, Diving Deeper

- **The Release Pipeline Model** - Michael Greene, Steven Murawski
- **Building a Simple Release Pipeline in PowerShell Using psake, Pester, and PSDeploy** - Brandon Olin
- **Stack Overflow: How We Do Deployment - 2016 Edition** - Nick Craver
- **DevOps Reading List** - Steven Murawski
- **Reading List** - Chris Hunt
- **The Pester Pipeline** - Chris Hunt
- **Best Practices with Packer and Windows** - Matt Hodgkins
- **Introduction to Kitchen-DSC** - Gael Colas
- **Testing Ansible Roles Against Windows with Test-Kitchen** - Matt Hodgkins
- Twitter, [Slack](https://slack.com), and other communities
- Etc.