Automated Site Provisioning

Site Designs
Provisioning Templates
PnP-PowerShell
Flow
Azure Functions

About Tom Castiglia…

SharePoint / Office 365 Consultant

SharePoint Saturday Speaker

President of the San Diego SharePoint User Group

Nintex vTE (virtual Technical Evangelist)

https://**docfluix.com**

meetup.com/**SoCalO365**/

tom@docfluix.com

slideshare.net/**tomcastiglia**

linkedin.com/in/**tomcastiglia**

@TomCastiglia

Agenda – Modern Site Provisioning

DocFluix

Component Overview

Tips & Tricks

Demos

NINTEX VTE

Component Overview

DocFluix

PnP-Powershell	Microsoft Flow	Azure
• Site Scripts / Site Designs • Provisioning Templates	• Http Trigger • Azure Queues Connector	• Storage Queues • Functions

NINTEX VTE

Pre-Requisites

- This session assumes you have some basic knowledge of…
  - PnP Powershell
  - Microsoft Flow
  - Azure
    - Functions
    - Storage Queues

Approaches

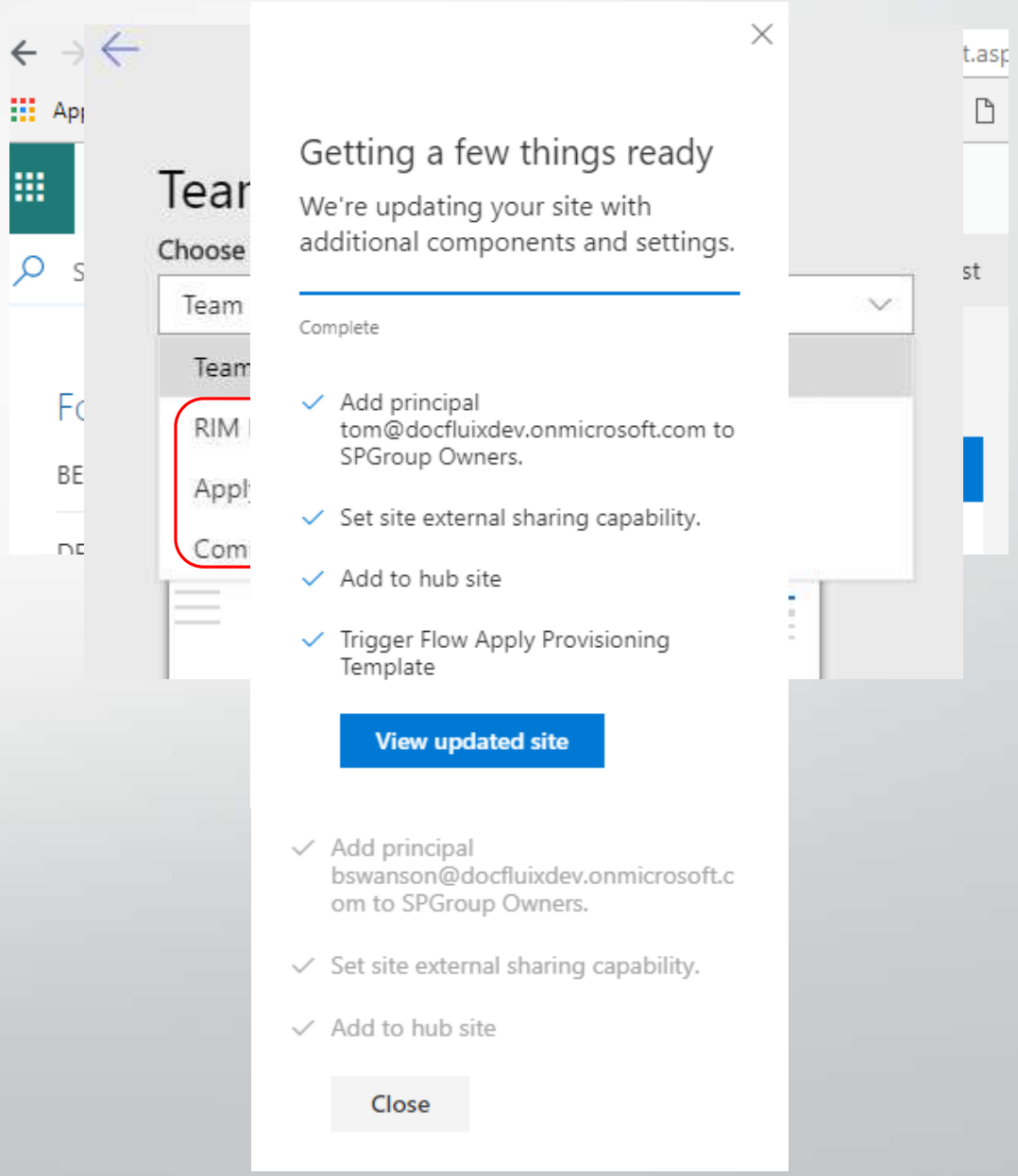
Ungoverned Self-Service Site Creation

Governed / Admin-Only Site Creation

Governed Self-Service Site Creation

Ungoverned Self-Service Site Creation

- Users can create sites through the Create Site button on SharePoint home page
- Optionally select a custom site design which may or may not invoke provisioning templates.
- Makes it easy for users to select custom designs and template
- Allows site sprawl due to lack of governance



Governed / Admin-Only Site creation

- Same model as self-service, but the **Create Site** button is not available in SharePoint home page.
- Admins must create new sites through Powershell or through SharePoint Admin Center.
- Otherwise, user experience for admins is identical to normal self-service Create Site

Governed Self-Service Site Creation

- Create Site button is disabled in SharePoint home
- Custom List is used to drive user requests for a new site
- Invoke Flow for approval by admin before creating a site
- Flow Process
  - Approval
  - Add Message to Azure Storage Queue
  - Initiates

Using Site Design to Invoke Provisioning Templates

- Site Designs can be used for…
  - Create a new SharePoint list
  - Define a new site column
  - Define a new content type
  - Add a navigation link
  - Remove a navigation link
  - Apply a theme
  - Set a site logo
  - Join a hub site
  - Install an add-in or solution
  - Register an extension
  - Trigger a flow
  - Configure regional settings
  - Add users (principals) to SharePoint Groups
  - Manage guest access

```
1  {
2      "$schema": "schema.json",
3      "actions": [
4          {
5              "verb": "setSiteExternalSharingCapability",
6              "capability": "ExternalUserSharingOnly"
7          },
8          {
9              "verb": "joinHubSite",
10             "hubSiteId": "11cd1697-a7a2-461d-aab1-84c8db273bff"
11         },
12         {
13             "verb": "triggerFlow",
14             "url": "https://prod-47.westus.logic.azure.com:443/wor
15             "name": "Apply Provisioning Template",
16             "parameters": {
17                 "projectType": "Renewables Project"
18             }
19         }
20     ],
21     "bindata": { },
22     "version": 1
23 }
```

Site Designs can't do everything…

- Cannot configure pages and web parts
- Cannot self-generate from an existing site
- Limited to 30 actions / sub-actions

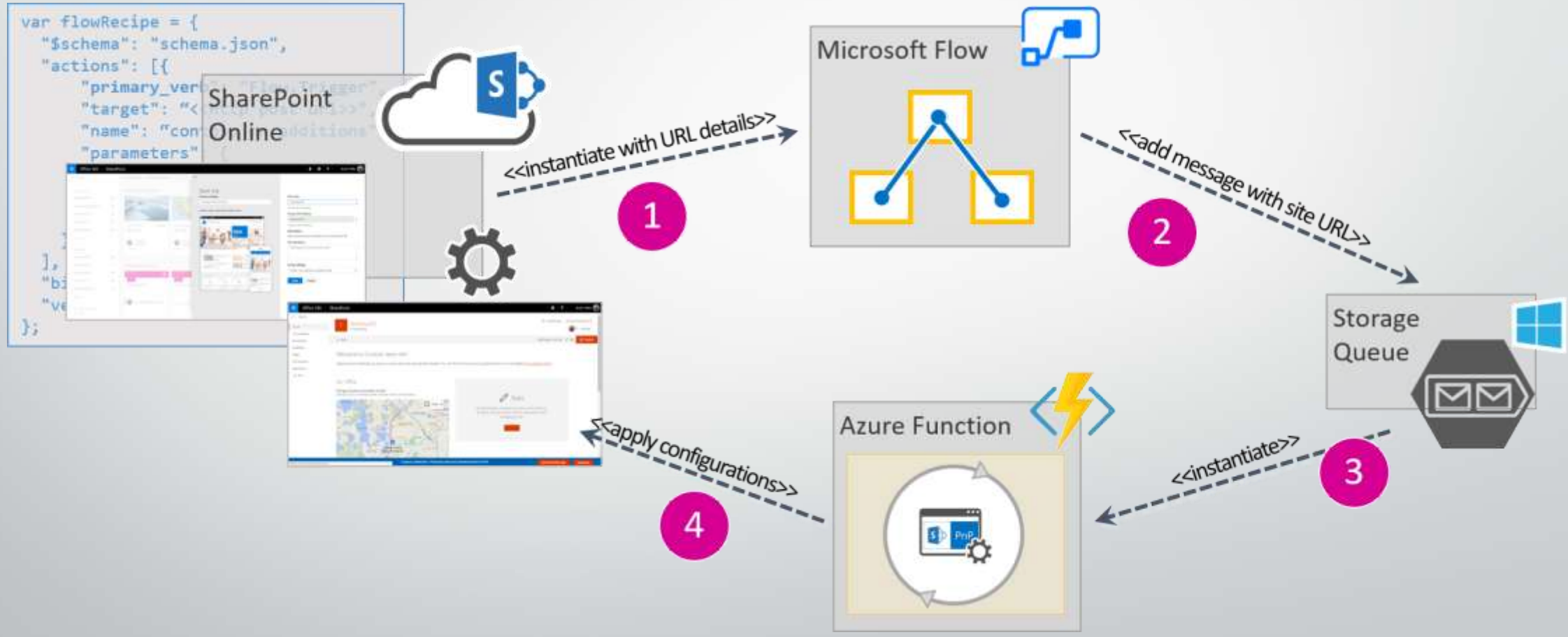
Build Provisioning Solution

- Setup App-Only Access (Register AppId & AppSecret)
- Create Azure Storage Queue
- Create Flow
  - Trigger: **When HTTP Request is received**
  - Action: **Azure - Put Message on a queue**
- Get Provisioning template
  - `Get-PnPProvisioningTemplate -out “IA-Template.pnp" -Handlers "Fields,ContentTypes,Lists,SiteSecurity" -IncludeSiteGroups`
  - `Get-PnPProvisioningTemplate -out "Pages-Template.pnp" -Handlers "Pages,PageContents,Navigation"`

Build Provisioning Solution (Con't)

- Create Azure Function
  - Powershell Queue Trigger
    - `Connect-PnPOnline -AppId $env:SPO_AppId -AppSecret $env:SPO_AppSecret -Url $url -ErrorAction Stop`
    - `Apply-PnPProvisioningTemplate -Path $templatePath -Handlers SiteSecurity -ErrorAction Stop`
  - Upload PnP Powershell Module to Azure
    - `Save-Module -Name SharePointPnPPowerShellOnline -Path [pathtoyourfolder]`
- Create Site Design
  - Include **triggerFlow** action to send Http Request to Flow

Provisioning Process



Basic Flow Configuration

When a HTTP request is received

HTTP POST URL [redacted]

Request Body JSON Schema

```
{
    "type": "object",
    "properties": {
        "webUrl": {
            "type": "string"
        },
        "parameters": {
            "type": "object",
            "properties": {
```

Use sample payload to generate schema

Show advanced options

Put a Fields Site Template message on newsitequeue

* Queue Name: newsitequeue

* Message:

```
{
    "SiteUrl": " webUrl ",
    "TemplateToApply": "[redacted]",
    "Description": "Fields Template",
    "TemplateType": "Fields"
}
```

Azure Function - Powershell

JSON Input Message from Queue

```
{
    "SiteUrl": "[your new site's url]",
    "TemplateToApply": "Your-Template.pnp",
    "Description": "[optional description]",
    "TemplateType":"Fields"
}
```

```
52    switch ($templateType)
53    {
54        "Fields"
55        {
56            $nextTemplateEntity = [PSObject]@{
57                SiteUrl = $url
58                TemplateToApply = "BE[redacted]"
59                Description = "ContentTypes Template"
60                TemplateType = "ContentTypes"
```

```
114     if ($nextTemplateEntity)
115     {
116         $nextTemplateEntity | ConvertTo-Json | Out-File -Encoding UTF8 $nextTemplateMsg
117     }
118                                                                                mplateDescription)"
119 }
120 catch
121 {
122     Write-Output "Exception Occured"
123     Write-Output "Exception Type: $($_.Exception.GetType().FullName)"
124     Write-Output "Exception Message: $($_.Exception.Message)"
125 }
126
```

```
76                TemplateToApply = "[redacted]"                                  "[redacted]s/"
77                Description = "FieldDocs Template"
78                TemplateType = "FieldDocs"
79            }                                                        "StepCompleted"="$templateType"}
80        }
```

Example Project Team Site

Columns & Content Types

- 10 custom site columns
- 15 custom content types (9 of which have custom templates)

Libraries

- 4 Document Libraries and 1 Picture library
- 3 of these libraries use 12 of the content types and all 10 columns
- About 100 standard documents deployed across three libraries (created via Flow rather after PnP Provisioning)
- One library has unique permissions
- One library is used to store templates referenced by content types
- Multiple Views

Custom homepage

- One Highlighted Content web part
- One Library web part
- Planner web part
- Left Navigation

Tip #1

DocFluix

Application Settings in Azure Functions

- Same concept as AppSettings in web.config or app.config
- Configurable through Azure Functions administration

Don’t name application settings with a name containing “Temp” or “Tmp”

- Azure will return the setting value as: **D:\local\copy** (regardless of the value that you store in the setting)

NINTEX VTE

Tip #2

PnP Powershell - **New-PnPSite…**

- this cannot authenticate using App Permissions.
- Requires auth via username and password (limitation in underlying SharePoint APIs)

Tip #3

Consider storing provisioning templates in a document library, and reference from Azure Functions Powershell

Optionally, you can upload to Azure Functions (but library makes updating templates easier)

Tip #3 (Con't)

DocFluix

When calling `Apply-PnPPowershell`

- If **-Path** param references a library instead of a local file (local to Azure),
- then template must be in "PnP" format instead of "Xml".

NINTEX VTE

Tip #4

Create modular Provisioning Templates

- If you execute this command:
  - `Get-PnPProvisioningTemplate -out "MyTemplate.xml"`
- It will create a giant template with every aspect of the site
- Instead create component templates, using Handlers like this
  - `Get-PnPProvisioningTemplate -out "IA-Template.xml" -Handlers "Fields,ContentTypes,Lists,SiteSecurity"`
  - `Get-PnPProvisioningTemplate -out "Pages-Template.xml" -Handlers "Pages,PageContents,Navigation"`
- Also, use Handlers when applying a template:
  - `Apply-PnPProvisioningTemplate -Path $templatePath -Handlers SiteSecurity -ErrorAction Stop`

PnP Provisioning Handlers

```
AuditSettings = 1,
ComposedLook = 2,
CustomActions = 4,
ExtensibilityProviders = 8,
Features = 16,
Fields = 32,
Files = 64,
Lists = 128,
Pages = 256,
Publishing = 512,
RegionalSettings = 1024,
SearchSettings = 2048,
```

```
SitePolicy = 4096,
SupportedUILanguages = 8192,
TermGroups = 16384,
Workflows = 32768,
SiteSecurity = 65536,
ContentTypes = 131072,
PropertyBagEntries = 262144,
PageContents = 524288,
WebSettings = 1048576,
Navigation = 2097152,
ImageRenditions = 4194304,
ApplicationLifecycleManagement = 8388608,
Tenant = 16777216,
WebApiPermissions = 33554432,
```

Tip #5

- With Azure Functions, if using “Consumption Plan” instead of “App Service Plan”, each function call has a default timeout of 5 minutes (Which can be increased to 10 minutes max).
- **`Apply-PnPProvisioningTemplate -Path “[pnp file url]”`**
  - A single large provisioning template can take more than 10 minutes
- Example –
  - “IA-Template.pnp” contains Fields, ContentTypes, Lists, SiteSecurity (took 8 – 15 minutes on average)
  - Site contains 12 content types, each of which are used in multiple libraries
  - Split IA-Template into four templates (Fields.pnp, ContentTypes.pnp, Lists.pnp & SiteSecurity.pnp)
  - Lists.pnp alone would occasionally timeout
  - Split Lists.pnp into three templates each one taking only 1 – 3 minutes (but 7 templates total).
  - Chained separate function calls for each template

Tip #6

- Having 7 templates is ***much too granular***.
  - Many dependencies between templates. Every change to a site requires tremendous regression testing.
- Switched Function App from Consumption Plan to App Service plan so that we could have longer time outs.
- Consolidated into 1 template with: **SiteSecurity, Fields, ContentTypes, Lists, Pages, PageContents, Navigation**

Tip #7

- You may wish to tweak a provisioning template after it is exported.
  - Export to XML (instead of PnP)
  - PnP is a zip file so you can rename the extension and review the contents
- Don't assume that template created from Get-PnPProvisioningTemplate will be perfect. Plan to tweak the exported Xml.
- Although you can create the provisioning template from scratch on your own, you can use Get-PnPProvisioningTemplate to create examples and snippets that you can use in manually generated templates
- Make your edits, then convert to PnP format:

```
$template = Read-PnPProvisioningTemplate -Path .\MyTemplate.xml
Save-PnPProvisioningTemplate -InputInstance $template -Out .\MyTemplate.pnp
```

Tip #8

- Problem - Occasional Race conditions
  - When using Apply-PnPProvisioningTemplate with a single template, I occasionally encountered unexplained errors:
    - **Invalid Field Name** (When adding a Field to a Content Type)
    - **The specified user Site Owners could not be found**. (When configuring unique permissions on library)
  - Running same template on same site again and it would work without error.

Tip #8 (Con't)

Solution –

- In Flow, Added 120 sec delay after site was created before placing item on queue
- Seemed to help but still saw occasional errors
- Modified Powershell to make multiple calls to Apply-PnPProvisioningTemplate with different Handlers, such as:

```
Apply-PnPProvisioningTemplate
-Path $templatePath
-Handlers SiteSecurity
-ErrorAction Stop
Start-Sleep -s 60

Apply-PnPProvisioningTemplate
-Path $templatePath
-Handlers Fields
-ErrorAction Stop
Start-Sleep -s 60
…
```

Tip #9

- **For Error Handling (try/catch) need to specify:**
  - `ErrorAction -Stop`
  - like this…

```
try {
    Apply-PnPProvisioningTemplate -Path $templatePath -ErrorAction Stop
}
catch {
    Write-Output "Error: $($_.Exception.GetType().FullName)"
    Write-Output "Msg: $($_.Exception.Message)"
}
```

Otherwise exceptions from Apply-PnPProvisioningTemplate won't hit your catch

Tip #10 - Leverage Provisioning Tokens

Token Names (small sample)	Return value
{siteid}	Returns id of the current site
{sitename}	Returns friendly name of the site (*example: I placed this token directly as the default value for a "ProjectName" site column, where each site's name reflects the project name.*)
{sitecollection}	Returns Server relative Url of the site collection
{fieldtitle:[internalname]}	Returns title of field by internal name (used when dealing with calculated columns)
{listid:[name]}	Returns a id of the list given its name
{listurl:[name]}	Returns a site relative url of the list given its name
{associatedmembersgroup} {associatedownersgroup} {associatedvisitorsgroup}	Returns the title of the associated member/owners/visitors SharePoint group of a site
{sitecollectiontermsetid:[termsetname]}	Returns the id of the given termset name located in the sitecollection termgroup

Other Tips

- Although PnP Provisioning allows us to store default content (e.g. documents, metadata, list items in a template, we decided not to do that).
- We are using a Hub site to store standard documents in various libraries.
- Use MS Flow to
  - Copy documents and metadata to each new site
  - Create Buckets and Tasks in Planner that are defined in lists in the Hub site
- This approach allows power users to easily change the default content and Planner items

Resources

- Provisioing Tokens:
  - https://github.com/SharePoint/PnP-Sites-Core/blob/master/Core/ProvisioningEngineTokens.md
- Overall solution:
  - https://docs.microsoft.com/en-us/sharepoint/dev/declarative-customization/site-design-pnp-provisioning