Navigating the shift to XM Cloud

Ronald van der Plas

Ronald van der Plas

Solution Architect

Content Hub Specialist

15+ years developer

MVP 2024
Technology

SITECORE

macaw

Where to start…?

Start at the beginning!

- Analyse the existing Sitecore environment
- Get XM Cloud and Vercel
- Downgrade XP to XM, rethink components
- Migrate to XM Cloud
- Create deployment pipelines
- Rebuild head
- Release, test and repeat

Analyse existing Sitecore environment

Our project starting point

- 50+ sites
- 100+ forms
- 24 Sitecore pipelines
- 100+ components

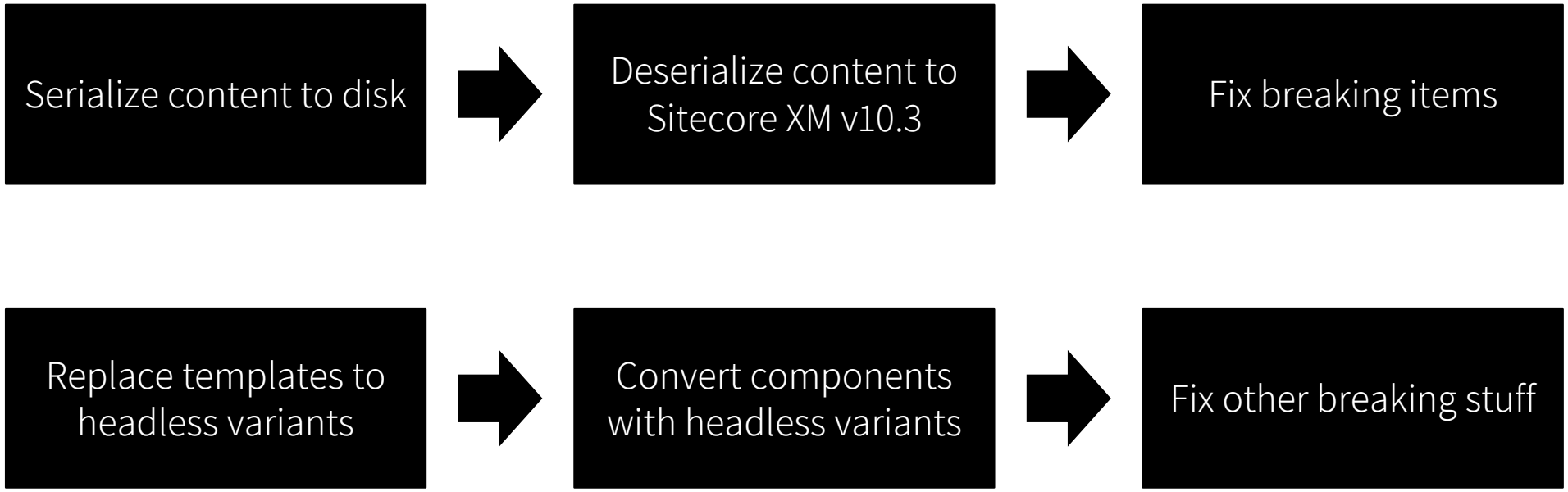
Don’t forget about

- Email Experience Manager (EXM)
- Search
- Lots of content, including media
- External API’s


Start with a major clean-up

Content migation

Migrate your content



How to migrate?

- Copy Production databases
- Restore databases withon local Docker instance for Sitecore XM v10.3*
- Further serialization with SCS -> *.yaml files
- Deserialize to Sitecore Headless
- Create a Macawsome Powershell script
  - Find and replace non-headless GUIDs to headless GUIDs
  - Templates, components, etc.

* Compatible version with XM Cloud

Shift into Forms

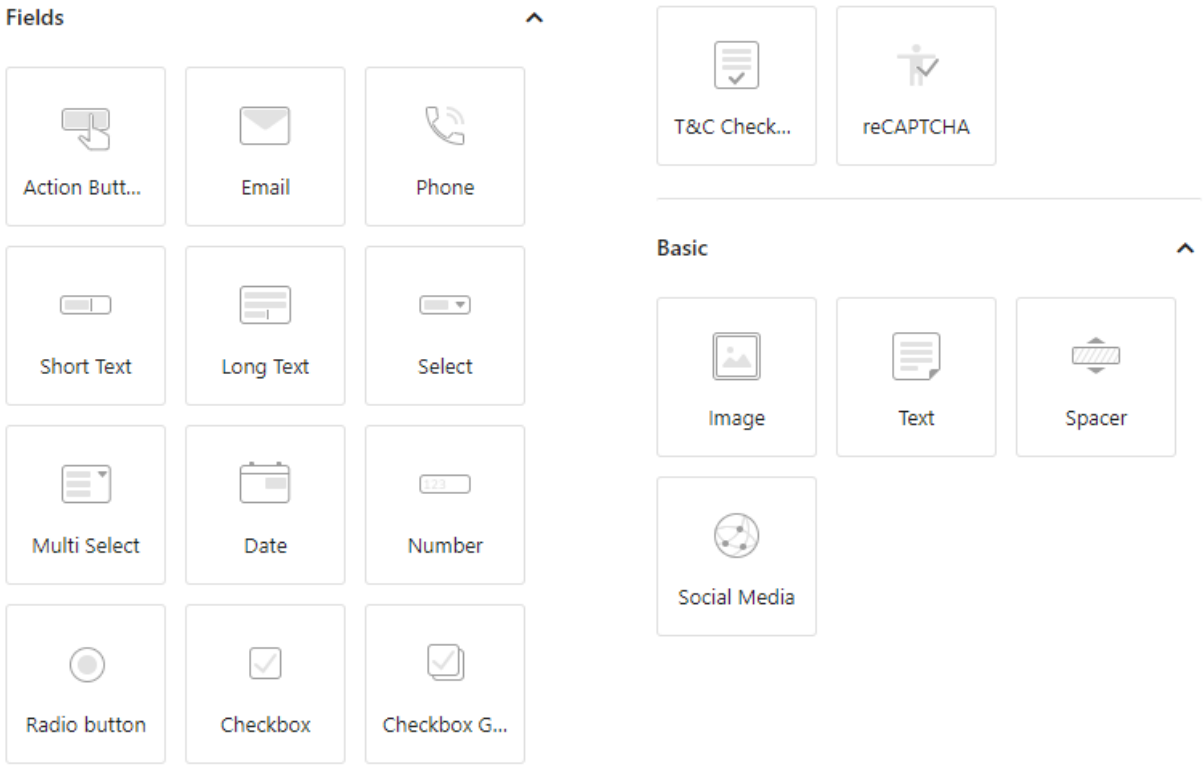
What about forms?!

- 100+ forms to migrate
- Some highly complex, due to sections, many conditions
- Broken links
- Incorrect fieldtypes
- Incorrect or unavailable validators

Would Sitecore be a good fit?

We say not (yet)

- Limited functionality
- Missing sections
- Missing custom components / actions



react-jsonschema-form to the rescue

react-jsonschema-form

✔ JSONSchema

```
{
  "title": "A registration form",
  "description": "A simple form example.",
  "type": "object",
  "required": [
    "firstName",
    "lastName"
  ],
  "properties": {
    "firstName": {
      "type": "string",
      "title": "First name"
    },
    "lastName": {
      "type": "string",
```

✔ UISchema

```
{
  "firstName": {
    "ui:autofocus": true
  },
  "age": {
    "ui:widget": "updown"
  },
  "bio": {
    "ui:widget": "textarea"
  },
  "password": {
    "ui:widget": "password",
    "ui:help": "Hint: Make it strong!"
  },
```

✔ formData

```
{
  "firstName": "Chuck",
  "lastName": "Norris",
  "age": 75,
  "bio": "Roundhouse kicking asses since 1940",
  "password": "noneed"
}
```

A registration form

A simple form example.

First name*

Chuck

Last name*

Norris

Age

75

Bio

Roundhouse kicking asses since 1940

Password

••••••

Hint: Make it strong!

Submit

react-jsonschema-form

Pros:

- Easy schema, highly flexible
- Free of charge
- Transformation from Sitecore Forms to JSON

Cons:

- JSON only, no easy WYSIWYG editor

Things to consider

- Simplefy overly complex forms
- Declutter the forms
- Convert into react-jsonschema-form
- Move existing submit actions to Azure Functions
- Save, test, repeat…

Next topic: EMX

What about Emails?!

- Sitecore offers Sitecore Send
- Customer wants to use Salesforce
- Temporary workaround use Sendgrid

Things to consider

- Reusing existing e-mail templates
- Clean-up unused email and templates
- Post-submit actions
- Use Sitecore Webhooks

What about the Head?

Rebuild head

- 100+ components
- Lift and shift approach
- Wrap existing React code to comply with XM Cloud
- Rebuild couple of razor views

How do we deploy all this?

Deployment pipelines

- Build and deploy from Azure DevOps
- Use them CLIs!


Show me the ~~money~~ pipelines

Vercel - Example

```yaml
steps:
Settings
- task: NodeTool@0
  inputs:
    versionSpec: '18.15.0'
  displayName: 'Install Node.js'

- script: |
    npm ci
    npm ci vercel
  workingDirectory: $(Build.SourcesDirectory)/src/sxastarter
  displayName: 'npm install'

- script: |
    vercel pull --yes --environment=preview --token=$(VERCEL_TOKEN)
  workingDirectory: $(Build.SourcesDirectory)/src/sxastarter
  displayName: 'Vercel | Pull environment'

- script: |
    vercel build --token=$(VERCEL_TOKEN)
  workingDirectory: $(Build.SourcesDirectory)/src/sxastarter
  displayName: 'Vercel | Build'

- script: |
    vercel deploy --prebuilt --token=$(VERCEL_TOKEN)
  workingDirectory: $(Build.SourcesDirectory)/src/sxastarter
  displayName: 'Vercel | Deploy'
```

Sitecore – Example *

```yaml
parameters:
  - name: xmcClientId
  - name: xmcClientSecret

steps:
  - powershell: |
      dotnet sitecore cloud login --client-credentials --client-id $env:XMC_CLIENT_ID --client-secret $env:XMC_CLIENT_SECRET --allow-write
    displayName: 'Login to Sitecore Cloud'
    env:
      XMC_CLIENT_ID: ${{ parameters.xmcClientId }}
      XMC_CLIENT_SECRET: ${{ parameters.xmcClientSecret }}
```

* Check out Rob Habraken's blog

Create deployment

```yaml
stages:
- stage: ${{ parameters.stageName }}
  jobs:
    - deployment: ${{ parameters.stageName }}_job
      variables:
        - group: 'sitecore-xm-cloud-variables'
      pool:
        vmImage: windows-latest
      environment: ${{ parameters.env }}
      strategy:
        runOnce:
          deploy:
            steps:
            - checkout: self

            - script: |
                dotnet tool restore
              displayName: dotnet tool restore

            - template: ../steps/login-to-sitecore-cloud.yaml
              parameters:
                xmcClientId: $(xmc-client-id)
                xmcClientSecret: $(xmc-client-secret)

            - powershell: |
                $deployment = dotnet sitecore cloud deployment create --environment-id $env:XMC_ENVIRONMENT_ID --no-watch --no-start --upload --working-dir $(Build.SourcesDirectory) --json | ConvertFrom-Json
                Write-Host "$deployment"
                $deploymentId = $deployment.id
                Write-Host "##vso[task.setvariable variable=deploymentId]$deploymentId"
                Write-Host "##vso[task.setvariable variable=deploymentId;isOutput=true]$deploymentId"
              name: CreateDeployment
              displayName: 'Create deployment'
              env:
                XMC_ENVIRONMENT_ID: $(xmc-${{ parameters.env }}-environment-id)

            - powershell: |
                $deploymentResult = dotnet sitecore cloud deployment start --deployment-id $(deploymentId) --json | ConvertFrom-Json
                if($deploymentResult.IsCompleted) {
                    exit 0
                }
                else {
                    Write-Host "$deploymentResult"
                    exit 1
                }
              displayName: 'Start deployment'
```

Promote deployment

```yaml
stages:
- stage: ${{ parameters.stageName }}
  dependsOn: ${{ parameters.dependsOn }}
  jobs:
    - deployment: ${{ parameters.stageName }}_job
      variables:
        - group: 'sitecore-xm-cloud-variables'
        - name: deploymentId
          value: $[stageDependencies.create_deployment_on_dev.create_deployment_on_dev_job.outputs['create_deployment_on_dev_job.CreateDeployment.deploymentId']]
      pool:
        vmImage: windows-latest
      environment: ${{ parameters.env }}
      strategy:
        runOnce:
          deploy:
            steps:
            - checkout: self

            - script: |
                dotnet tool restore
              displayName: dotnet tool restore

            - template: ../steps/login-to-sitecore-cloud.yaml
              parameters:
                xmcClientId: $(xmc-client-id)
                xmcClientSecret: $(xmc-client-secret)

            - powershell: |
                dotnet sitecore cloud environment promote --environment-id $env:XMC_ENVIRONMENT_ID --source-id $(deploymentId)
              displayName: 'Promote deployment'
              env:
                XMC_ENVIRONMENT_ID: $(xmc-${{ parameters.env }}-environment-id)
```

Things to consider

- Rethink your Git strategy
- Sitecore and Vercel work with promoting environments
- Finally leverage CI/CD!
- Test approved? -> Promote environment to next state, repeat until production
- Think about your content (including templates)

Lessons learned

- Clean up content before migration
- Create a repeatable content migration
- Redesign your deployment cycle
- Simplify existing forms before migration
- Components are sometimes way more complex than anyone can remember. Do a proper analysis
- Don’t try to migrate everything at once. Consider a site-by-site migration


Any questions?

Special thanks to the team!

Gary
Migration magician

Arno
Migration buddy

Erwin
Wrapmaster

Peter
Forms wizard

Sjoerd
Platform expert

Thank you!