ITS Documentation

Low-Level Design Document – V1.1

Table of Contents

[1. Introduction 3](#_Toc128731258)

[1.1 Design Scope 3](#_Toc128731259)

[1.2 Abbreviations & Definitions 3](#_Toc128731260)

[1.3 References 5](#_Toc128731261)

[Markdown syntax for wikis 5](#_Toc128731262)

[Syntax guidance for basic Markdown usage 5](#_Toc128731263)

[Markdown tools 5](#_Toc128731264)

[Azure DevOps Integration Documentation Project 5](#_Toc128731265)

[Integration Documentation Web Site 5](#_Toc128731266)

[GitHub repo for Jekyll theme 1 5](#_Toc128731267)

[GitHub repo for Jekyll theme 2 5](#_Toc128731268)

[Step-by-step Ruby & Jekyll configuration 5](#_Toc128731269)

[Deploy as a static web site 5](#_Toc128731270)

[Build the pipeline for deployment 5](#_Toc128731271)

[2. System Overview 6](#_Toc128731272)

[3. Detailed System Design 6](#_Toc128731273)

[3.1 How to create documentation 6](#_Toc128731274)

[Syntax guidance for basic Markdown usage 7](#_Toc128731275)

[3.2 How to build and test the documentation on the local computer 8](#_Toc128731276)

[3.3 How to build the Azure DevOps documentation project’s pipeline 8](#_Toc128731277)

[3.4 How to deploy the documentation project 10](#_Toc128731278)

[3.5 How to secure the documentation web site 14](#_Toc128731279)

[3.6 How to create a central documentation project and build a pipeline using individual projects content. 15](#_Toc128731280)

[3.7 How to create and publish documentation with Nodinite 20](#_Toc128731281)

[4. Approvals 25](#_Toc128731282)

Table of Figures

[Figure 1 - Markdown documentation sample 7](#_Toc128732725)

[Figure 2 - VS Code displaying markdown file side-by-side. 8](#_Toc128732726)

[Figure 3 - Documentation project's web site 10](#_Toc128732727)

[Figure 4 - Project folder's structure 10](#_Toc128732728)

[Figure 5 - Set up the page's title 11](#_Toc128732729)

[Figure 6 - Web site main's page content 11](#_Toc128732730)

[Figure 7 - Publish the documentation project as a Wiki site. 12](#_Toc128732731)

[Figure 8 - Documentation display inside the project as a Wiki page 12](#_Toc128732732)

[Figure 9 - Select the existing pipeline option. 13](#_Toc128732733)

[Figure 10 - Create the release deployment. 14](#_Toc128732734)

[Figure 11 - Delete the script files used by the build. 15](#_Toc128732735)

[Figure 12 - Deploy release to the web site. 15](#_Toc128732736)

[Figure 13 - Role permissions added to the site. 16](#_Toc128732737)

[Figure 14 - Centralized project that builds the central web site. 17](#_Toc128732738)

[Figure 15 - Merge individual projects content 17](#_Toc128732739)

[Figure 16 - Configuration settings to access individual project when running the pipeline. 18](#_Toc128732740)

[Figure 17 - Merge project's content in the centralized repo 21](#_Toc128732741)

[Figure 18 - Nodinite documentation 22](#_Toc128732742)

[Figure 19 - Copy markdown files to the build folder. 22](#_Toc128732743)

[Figure 20 - Pipeline's script task that pushes documentation via an API. 23](#_Toc128732744)

**Document Version Control**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document Version | Action | Author | Date | Changes |
| 1.0 | Create | Eugen Frunza |  |  |
|  |  |  |  |  |

**Reviewers**

|  |  |
| --- | --- |
| Name | Title |
|  |  |
|  |  |

# Introduction

This design document describes the technical details required to create and deploy documentation using markdown.

## Design Scope

The design document provides technical details about the following areas:

* How to use markdown and create documentation
* How to build and test the documentation on the local computer
* How to build the Azure DevOps documentation project’s pipeline
* How to deploy the documentation project
* How to secure the documentation web site
* How to create a central documentation project and build a pipeline using individual projects content.
* How to create and publish documentation with Nodinite

## Abbreviations & Definitions

Table 2: Abbreviations and definitions

|  |  |
| --- | --- |
| **Abbreviation** | **Definition** |
| Markdown | A lightweight markup language that you can use to add formatting elements to plaintext text documents |
| Ruby | A dynamic, open-source programming language with a focus on simplicity and productivity. |
| Jekyll | Ruby script for building static web sites from simple HTML and markdown content using templates and add-ons. |

## 

## References

Table 3: References

|  |  |
| --- | --- |
| **Reference** | **Location** |
| High-Level Design Document | [ITS Central Documentation HLD.docx](https://bowvalley.sharepoint.com/:w:/s/DevelopmentTeam/EaUO5weX29VEsD7Rtk8cakwBwYRyqO3LNKcCjl8Q0pq0tw?e=6Q3iyz) |
| [Markdown syntax for wikis](#_Markdown_syntax_for) | <https://learn.microsoft.com/en-us/azure/devops/project/wiki/wiki-markdown-guidance?source=recommendations&view=azure-devops> |
| [Syntax guidance for basic Markdown usage](#_Syntax_guidance_for) | <https://learn.microsoft.com/en-us/azure/devops/project/wiki/markdown-guidance?view=azure-devops> |
| [Markdown tools](#_Markdown_tools) | <https://www.markdownguide.org/tools/> |
| [Azure DevOps Integration Documentation Project](#_Azure_DevOps_Integration) |  |
| [Integration Documentation Web Site](#_Integration_Documentation_Web) | <https://wa-integrationdocs-dev-001.azurewebsites.net/> |
| [GitHub repo for Jekyll theme 1](#_GitHub_repo_for) | <https://github.com/jasonlong/architect-theme> |
| [GitHub repo for Jekyll theme 2](#_GitHub_repo_for_1) | <https://github.com/pietromenna/jekyll-architect-theme> |
| [Step-by-step Ruby & Jekyll configuration](#_Step-by-step_Ruby_&) | <https://jekyllrb.com/docs/installation/windows/> |
| [Deploy as a static web site](#_Deploy_as_a) | [https://gunnarpeipman.com/jekyll-azure-static-website**/**](https://gunnarpeipman.com/jekyll-azure-static-website/) |
| [Build the pipeline for deployment.](#_Build_the_pipeline) | <https://gunnarpeipman.com/jekyll-azure-devops-build-pipeline/> |

# System Overview

Please read the “ITS Documentation High Level Design Document” which provides more details about the architectural overview and design decisions regarding this system.

# Detailed System Design

## How to create documentation

The documentation project will contain a collection of “md” files and associated artifacts (images, videos, etc.) that will be rendered to html files in the published site.

We can create documentation per project in Azure DevOps or a central documentation project to be maintained by all the developers.

The documentation can format the text nicely (Table of Contents, lists, bullet points, various formats) and include diagrams (sequence, graph, Gantt charts, journeys, etc.), images, videos, embedded Swagger Editor and so on.

A picture containing graphical user interface

Description automatically generated

Figure 1 - Markdown documentation sample

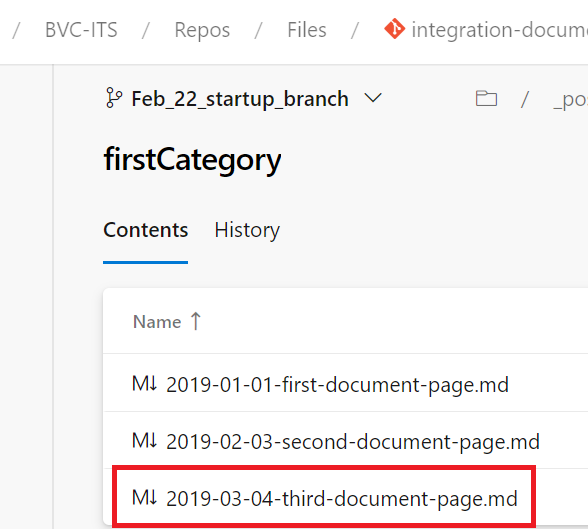
The following links explains how to write and format the text and add diagrams, images, and videos:

[Markdown syntax for wikis](#_Markdown_syntax_for)

# [Syntax guidance for basic Markdown usage](#_Syntax_guidance_for)

I have created one file containing most of these formats and added it to the repo from below:

<https://dev.azure.com/bowvalleycollege/BVC-ITS/_git/integration-documentation>



You can open this file with VS Code.

Text

Description automatically generated

Figure 2 - VS Code displaying markdown file side-by-side.

By clicking the first icon marked with the red rectangle, you can see the readme file code and the preview of it, side-by-side.

You can open the Preview as well by right-clicking on the file and selecting: Open Preview

## How to build and test the documentation on the local computer

There are several tools available to build the markdown files/project and build the html static site. You can find more information in the link from below:

[Markdown tools](#_Markdown_tools)

Each of these tools uses a different runtime to write the project and build the local environment / site (Ruby, Python, NodeJs).

Each tool uses a different UI format (theme) to display the web site’s content.

The themes for the Jekyll project are to be found here:

[GitHub repo for Jekyll theme 1](#_GitHub_repo_for)

[GitHub repo for Jekyll theme 2](#_GitHub_repo_for_1)

I have used to code the documentation web site using Ruby and run with Jekyll the scripts to build the project.

To run the site locally, you must install Ruby and Jekyll first, as described below:

[Step-by-step Ruby & Jekyll configuration](#_Step-by-step_Ruby_&)

With Ruby installed, install Jekyll from the terminal:

gem install jekyll bundler

## How to build the Azure DevOps documentation project’s pipeline

Download the project from the repository and open it in VS Code.

Execute the following 2 commands into the VS Code Terminal:

Jekyll build

Jekyll serve

The web site will start locally at this location: <http://localhost:4000>

Graphical user interface, text, application

Description automatically generated

Figure 3 - Documentation project's web site

The Documentation project’s structure can be seen below:

**Text

Description automatically generated**

Figure 4 - Project folder's structure

The main documentation files, having the extension “md”, are in the “\_posts” folder.

You can create a new sub-folder under the “\_posts” folder and add your documentation file there.

Keep in mind that the documentation file’s title that is displayed in the main web site’s page it is set up at the beginning of the file, in the following section:

Text

Description automatically generated

Figure 5 - Set up the page's title

Graphical user interface, text, application

Description automatically generated

Figure 6 - Web site main's page content

To understand the web site better, you can open the “\_layouts\default.html” file and see from where the other elements are incorporated into the main page.

Any changes you are making to this project will be built automatically and the web site refreshed while the server is up and running.

## How to deploy the documentation project

There are 3 ways in which the documentation can be published:

* + 1. Azure Wiki pages

Upload the documentation files into your repo,

Associate the repo to the Azure DevOps project and

Publish the repo as a Wiki:

Graphical user interface, application

Description automatically generated

Figure 7 - Publish the documentation project as a Wiki site.

The documentation will be available as displayed below:

Graphical user interface, text, application, email

Description automatically generated

Figure 8 - Documentation display inside the project as a Wiki page

**Note** that Azure DevOps displays properly all the complex markdown existing in the file.

* + 1. Static web site in an Azure Storage Account

Details can be found here:

[Deploy as a static web site](#_Deploy_as_a)

[Build the pipeline for deployment](#_Build_the_pipeline)

The command to load the files into the Azure Storage Static Website is displayed below:

az storage blob upload-batch --account-name efrunzadocs -s ./ -d '$web'.

Permission can be configured for this static web site by creating shared access signature URI per user who requires access to the site.

* + 1. Html web site in an Azure App Service

The app service needs to be created ahead of time as describe here:

<https://learn.microsoft.com/en-us/azure/app-service/quickstart-html>

Create the build pipeline by following the next steps:

Graphical user interface, text, application, email

Description automatically generated

Figure 9 - Select the existing pipeline option.

Select the buildConfiguration.vsts-ci.yml

Run the pipeline to generate the build.

If you review the file, you will see that it does the following:

* Run a build agent on Ubuntu.
* Install the runtime for Ruby and Jekyll.
  + script: |

gem install jekyll bundler

bundle install --retry=3 --jobs=4

* Build the project.
  + script: |

bundle install

jekyll build

Publish the build artifact with the name of “site”.

Create the release deployment by following the next steps:

* Import the artifacts.

Graphical user interface, text, application, email

Description automatically generated

Figure 10 - Create the release deployment.

Add 2 build tasks:

Graphical user interface, text, application, email

Description automatically generated

Figure 11 - Delete the script files used by the build.

Graphical user interface, text, application, email

Description automatically generated

Figure 12 - Deploy release to the web site.

The documentation web site for the POC has been deployed here:

<https://wa-integrationdocs-dev-001.azurewebsites.net/>

## How to secure the documentation web site

The Azure DevOps project can be configured with various level of permissions, as for any Azure DevOps project:

Graphical user interface, application, Teams

Description automatically generated

Figure 13 - Role permissions added to the site.

## How to create a central documentation project and build a pipeline using individual projects content.

When centralizing the individual projects documentation, several alternatives have been identified in the High-Level Design document.

We are presenting below the technical details for the following scenario:

* Have each developer add documentation local to their ITS project they are working on.
* Create a central documentation project and build a pipeline which copies the relevant documentation from each individual project, build it and deployed it to the central documentation web site.

Diagram

Description automatically generated

Figure 14 - Centralized project that builds the central web site.

When centralizing the individual projects documentation, a certain folder structure will have to be respected by the developers for their individual project.

Chart, schematic, bar chart

Description automatically generated

Figure 15 - Merge individual projects content

The centralized project will need to have permission to access the individual projects to be able to check out the folders and files needed.

You will need to provide as well certain configuration settings at the DevOps project’s level to allow for the centralized pipeline to access the individual projects as shown below:

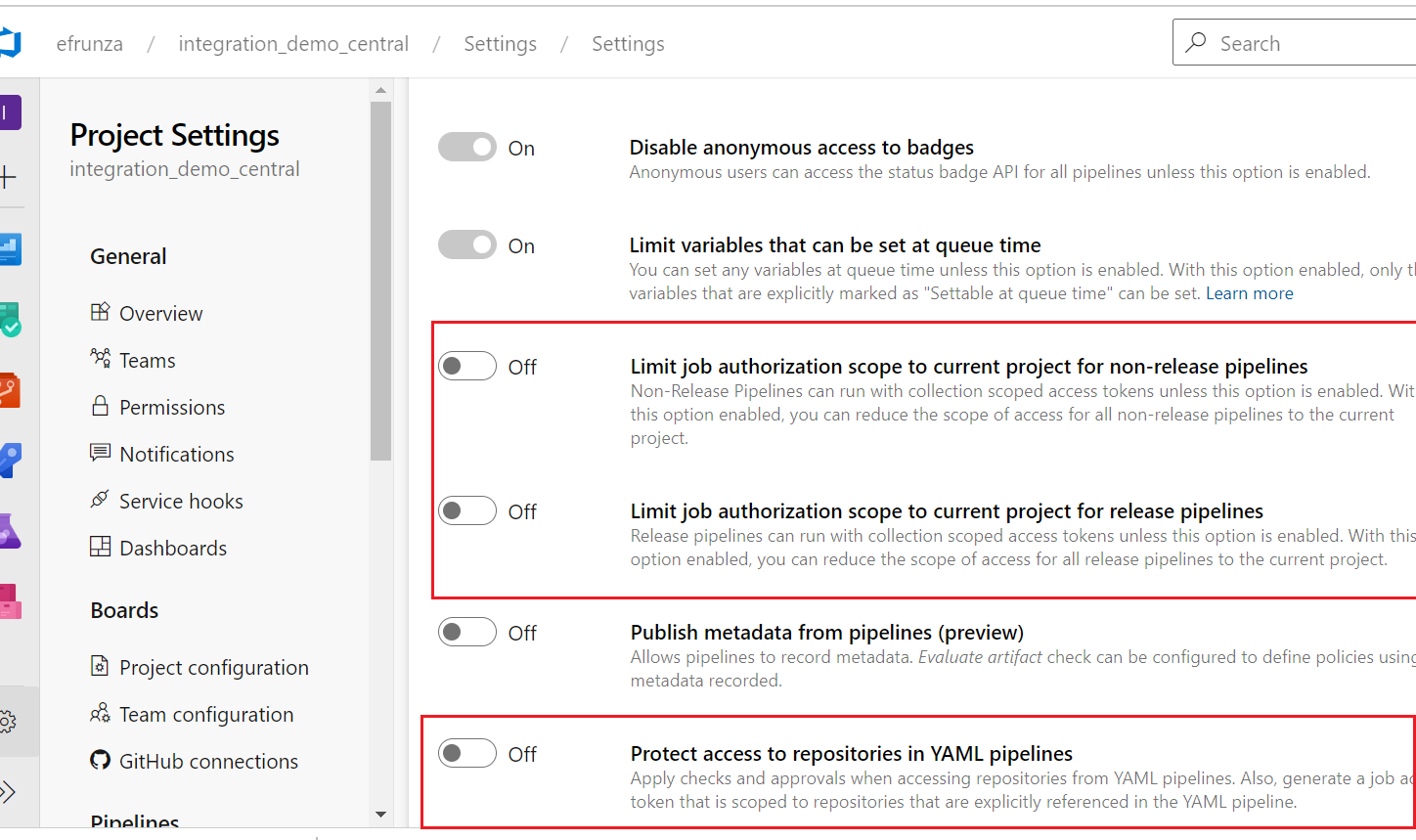


Figure 16 - Configuration settings to access individual project when running the pipeline.

The pipeline will be built as below (**add only the link to the pipeline’s task – it can be reviewed in there**):

# Starter pipeline

# Start with a minimal pipeline that you can customize to build and deploy your code.

# Add steps that build, run tests, deploy, and more:

# https://aka.ms/yaml

resources:

  repositories:

  - repository: integration\_demo\_project1

    type: git

    name: integration\_demo\_project1/\_git/integration\_demo\_project1

    path: $(Build.Repository.Name)

  - repository: integration\_demo\_project2

    type: git

    name: integration\_demo\_project2/\_git/integration\_demo\_project2

    path: $(Build.Repository.Name)

trigger:

- master

pool:

  vmImage: 'ubuntu-latest'

steps:

- task: UseRubyVersion@0

  inputs:

    versionSpec: '>= 2.5'

- checkout: self

  persistCredentials: true  # persists cred to perform some git remote commands like git push --tags

  path: $(Build.Repository.Name)

- checkout: integration\_demo\_project1

  path: integration\_demo\_project1

- checkout: integration\_demo\_project2

  path: integration\_demo\_project2

- script: |

    cp "$BUILD\_REPOSITORY\_LOCALPATH/." "$SYSTEM\_DEFAULTWORKINGDIRECTORY" -r

    echo "$SYSTEM\_DEFAULTWORKINGDIRECTORY"

    ls -a /home/vsts/work/1/s

    cp /home/vsts/work/1/integration\_demo\_project1/\_posts/\* "$SYSTEM\_DEFAULTWORKINGDIRECTORY/SourceRepo" -r

    cp /home/vsts/work/1/integration\_demo\_project2/\_posts/\* "$SYSTEM\_DEFAULTWORKINGDIRECTORY/SourceRepo" -r

    echo '%%% Copied sourcerepo %%%'

    ls -a "$SYSTEM\_DEFAULTWORKINGDIRECTORY/SourceRepo"

    echo '%%% Copied sourcerepo FirstCategoryProject1 %%%'

    ls -a "$SYSTEM\_DEFAULTWORKINGDIRECTORY/SourceRepo/firstCategoryProject1"

  # cp /home/vsts/work/1/integration\_demo\_project1/\_posts/\* /home/vsts/work/1/integration\_demo\_central/SourceRepo -r

  displayName: Copy Project1 content to SourceRepo folder

- script: |

    gem install jekyll bundler

    bundle install --retry=3 --jobs=4

    echo 'Build here'

    pwd

  displayName: 'jekyll'

- script: |

    pwd

    ls -a

  displayName: Log 1

- script: |

    echo '#### Build here ###'

    pwd

    jekyll build

  displayName: 'jekyll'

  condition:

- script: |

    pwd

    echo '#### We are here ###'

    ls -a

    echo '#### Files in \_site ###'

    ls -a /home/vsts/work/1/s/\_site

    echo '#### Files in \_site/SourceRepo ###'

    ls -a /home/vsts/work/1/s/\_site/SourceRepo

  displayName: Log 2

- task: CopyFiles@2

  displayName: 'Copy Files to: $(Build.ArtifactStagingDirectory)'

  inputs:

    SourceFolder: '\_site'

    TargetFolder: '$(Build.ArtifactStagingDirectory)'

- task: PublishBuildArtifacts@1

  inputs:

    pathtoPublish: '$(Build.ArtifactStagingDirectory)'

    artifactName: site

The end-result of this pipeline built is the project’s structure from below that will be deployed to the central web site.

We must spend a bit more time making sure the central web site UI layout displays properly all the individual projects and their documentation pages in the right menu items, sections, and categories.

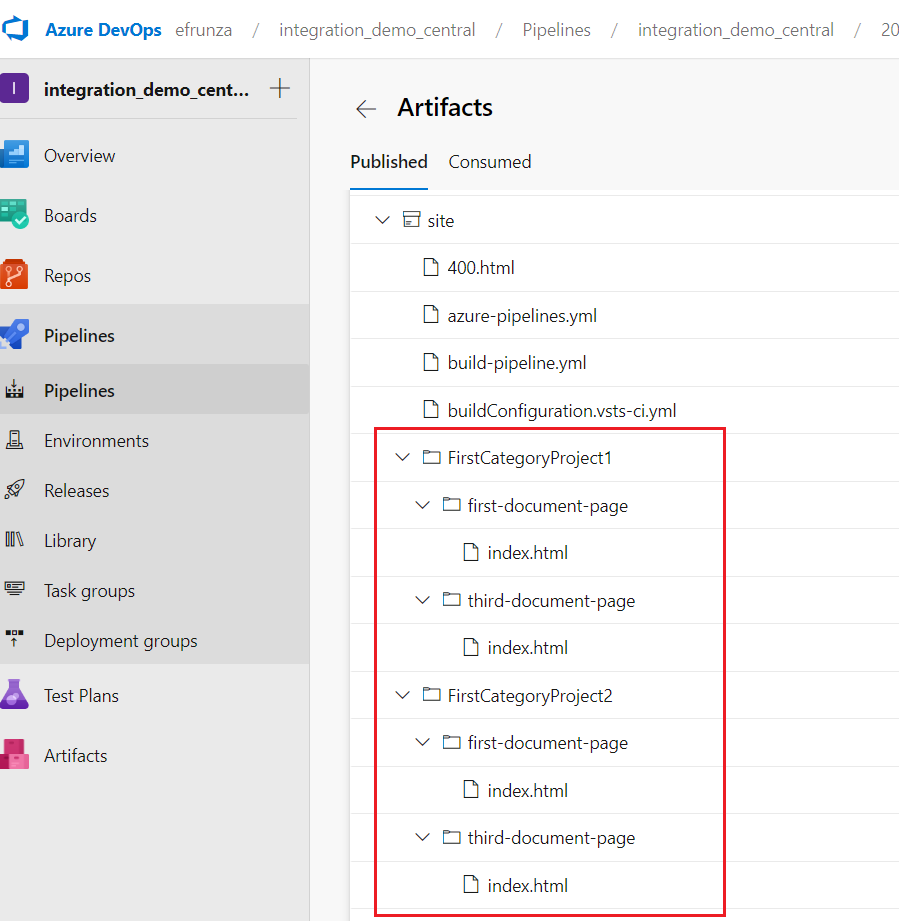


Figure 17 - Merge project's content in the centralized repo

## How to create and publish documentation with Nodinite

Documentation can be added in Nodinite as well, by creating a pipeline’s task that pushes all the markdown files to Nodinite via an API.

The documentation is created per project.

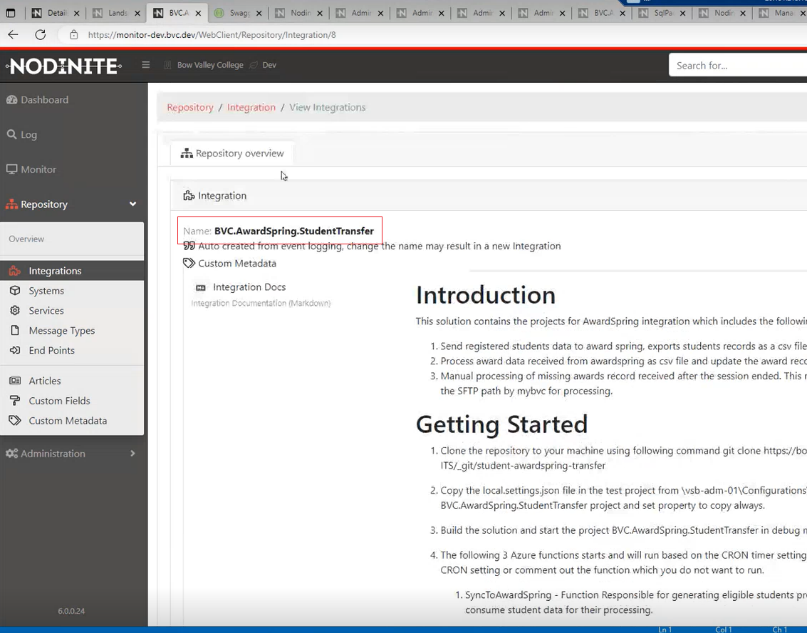


Figure 18 - Nodinite documentation

3.7.1 Set up the build pipeline to copy all the MD files into the pipeline’s target folder.

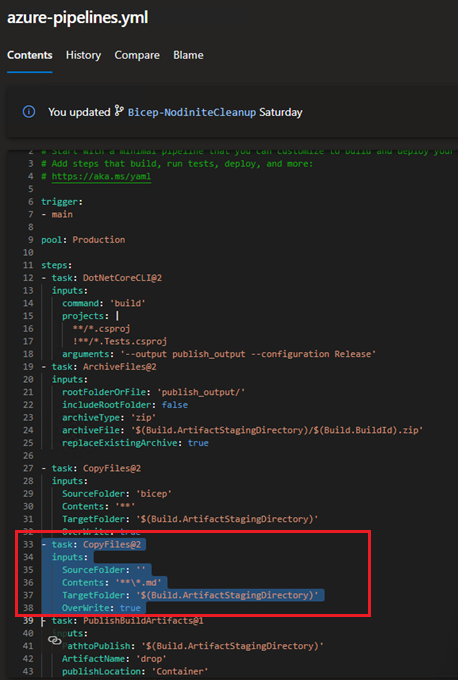


Figure 19 - Copy markdown files to the build folder.

3.7.2 In the Release pipeline, add a task to run an inline script to grab the markdown files and push them to Nodinite.

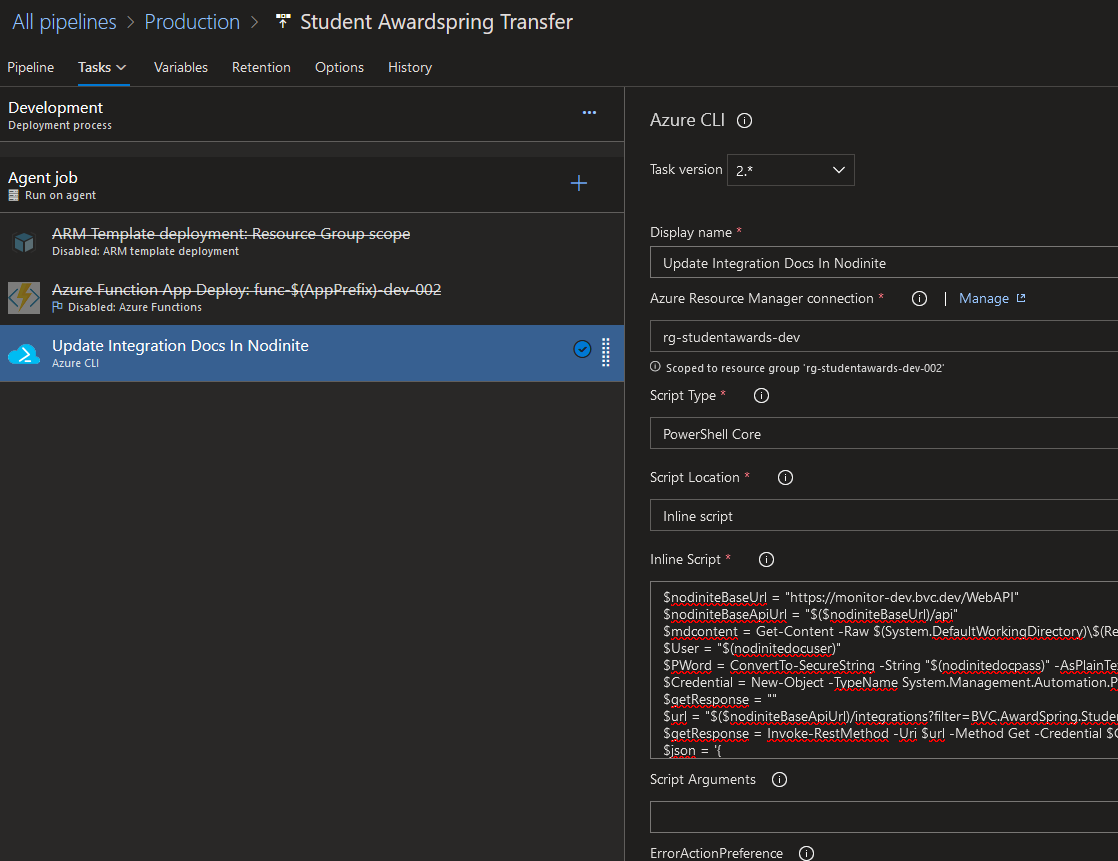


Figure 20 - Pipeline's script task that pushes documentation via an API.

The script content it is displayed below:

$nodiniteBaseUrl = "https://monitor-dev.bvc.dev/WebAPI"

$nodiniteBaseApiUrl = "$($nodiniteBaseUrl)/api"

$mdcontent = Get-Content -Raw $(System.DefaultWorkingDirectory)\$(Release.PRIMARYARTIFACTSOURCEALIAS)\drop\README.md

$User = "$(nodinitedocuser)"

$PWord = ConvertTo-SecureString -String "$(nodinitedocpass)" -AsPlainText -Force

$Credential = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $User, $PWord

$getResponse = ""

$url = "$($nodiniteBaseApiUrl)/integrations?filter=BVC.AwardSpring.StudentTransfer"

$getResponse = Invoke-RestMethod -Uri $url -Method Get -Credential $Credential

$json = '{

"CustomMetaDataId": 1,

"RepositoryTypes": 17,

"RepositoryTypesRequired": 0,

"DataType": 10,

"CustomValues": [

{

"Value": "## Chart \n<div class=\"mermaid\">\ngraph LR\n\tsubgraph \"Systems\"\n\t\tro[API Mgmt] ---roLA[Logic Apps]\n\t\troLA ---ro2[ERP]\n\tend\n</div>\n\n<div class=\"mermaid\"> \nclassDiagram\n class Duck{\n +String beakColor\n +swim()\n +quack()\n }\n class Fish{\n -int sizeInFeet\n -canEat()\n }\n class Zebra{\n +bool is\_wild\n +run()\n }\n Animal <|-- Duck\n Animal <|-- Fish\n Animal <|-- Zebra\n Animal : +int age\n Animal : +String gender\n Animal: +isMammal()\n Animal: +mate()\n</div> \n\*Example diagram\*\n<div class=\"alert alert-warning\">\nYou must always alert the the finance team before any deployments are performed \n</div>\n\n## Security classification\n\n>\*\*Internal\*\* - \*Do not share outside the organization\* \n\n<div class=\"mermaid\"> \nflowchart LR\n\tA o--o B\n\tB <--> C\n\tC x--x D\n\n</div>",

"Description": "",

"WebSite": "",

"FileBase64": null,

"ContentType": null,

"File": null,

"FileChanged": false,

"ValueType": null,

"Size": null,

"IsActive": true,

"IsDeleted": false,

"CreatedBy": "BOWVALLEY\\zngan",

"Created": "2023-02-22T22:19:23.5900761Z",

"ChangedBy": "BOWVALLEY\\zngan",

"Changed": "2023-02-24T15:55:09.1170482Z",

"Timestamp": null

}

],

"Name": "Integration Docs",

"Description": "Integration Documentation (Markdown)",

"WebSite": null,

"IsActive": true,

"IsDeleted": false,

"CreatedBy": "BOWVALLEY\\zngan",

"Created": "2023-02-22T22:13:40.6486497Z",

"ChangedBy": "BOWVALLEY\\zngan",

"Changed": "2023-02-25T04:59:11.1502468Z",

"Timestamp": "AAAAAAA9kFQ="

}'

$md = convertfrom-json $json

$md.CustomValues[0].Value = $mdcontent

$body = $getResponse.Collection.Items[0].Data

$body.CustomMetaDatas = @($md)

$getUpdateResponse = ""

$urlupdate = "$($nodiniteBaseApiUrl)/integrations/$($body.IntegrationId)"

$getUpdateResponse = Invoke-RestMethod -Uri $urlupdate -Method Put -Credential $Credential -Body ($body|ConvertTo-Json -Depth 25) -ContentType "application/json"

# Approvals