

Configuration du Prototype d'implémentation des tests d'accessibilité dans Azure Devops

Skills Evolution Program, CentraleSupélec, Microsoft

Jérôme Cornier – jcornier@microsoft.com

Avril 2019

Résumé



Les solutions numériques se multiplient et parfois sont le seul média permettant d'accéder à certaines informations ce qui rend nécessaire la prise en compte de l'accessibilité de ces solutions. La législation a évolué dans ce sens, rendant obligatoire la conformité des sites Web publics aux standards d'accessibilité depuis Septembre 2018.



La conjugaison des exigences d'accessibilité plus importantes et du nombre croissant de solutions amène à réfléchir sur l'automatisation des tests d'accessibilité dans un contexte DevOps pour améliorer en continue la qualité des produits.



Ce document accompagne un mémoire portant sur l'implémentation des pratiques inclusives dans un continuous delivery pipeline. Il décrit la mise en œuvre d'un prototype Azure DevOps utilisant un agent spécifique d'automatisation des tests d'accessibilité basés sur l'API de Deque University.

Table des matières

1	Introduction	5
1.1	Prérequis.....	5
2	Web App Classique - Préparation de l'environnement	6
2.1	Architecture générale.....	6
2.1.1	Architecture Azure de la Web App Windows Classique	6
2.1.2	Architecture du pipeline pour la Web App Classique	7
2.2	Création de l'application Web et publication dans GitHub.....	8
2.2.1	Création de l'application dans Visual Studio.....	8
2.2.2	Publication de l'application dans GitHub.....	10
2.2.3	Ajout d'une page Web dans l'application	12
2.2.4	Mise à jour dans GitHub.....	15
2.3	Création de l'environnement Azure	16
2.4	Création et configuration de l'organisation Azure DevOps.....	21
2.4.1	Création d'une nouvelle organisation DevOps	22
2.4.2	Création d'un nouveau Projet DevOps.....	22
2.4.3	Création du pipeline « Builds »	23
2.4.4	Création du pipeline « Release ».....	26
3	Création de l'Agent Azure DevOps spécifique A11y	31
3.1	Architecture Générale	31
3.1.1	Architecture Azure	31
3.1.2	Architecture de la machine virtuelle de l'Agent A11y	32
3.2	Création de la VM Azure pour l'Agent DevOps	32
3.3	Installation des dépendances logicielles	37
3.3.1	Configuration Internet.....	38
3.3.2	Installation de Firefox.....	38
3.3.3	Installation du pilote Gecko	40
3.4	Installation de l'agent DevOps.....	42
4	Implémentation de l'Agent DevOps A11y dans le pipeline	47
4.1	Implémentation de l'Agent dans le pipeline « Release »	47
4.2	Déploiement de l'application Web « Production »	51

4.3	Pipeline final	53
5	Démonstration des tests d'accessibilité automatisés	54
5.1	Première Release en échec avec la Web App initiale.....	54
5.1.1	Erreur n°1, paramètre de langue du document manquant	55
5.1.2	Erreur n°2, texte alternatif manquent sur une image	55
5.2	Mise en conformité du code d'après le rapport de tests.....	56
5.2.1	Correction du paramètre de langue.....	56
5.2.2	Correction du texte alternatif de l'image.....	56
5.2.3	Synchronisation des modifications avec GitHub.....	56
5.3	Release réussie après correction.....	57
5.4	Elévation du niveau d'accessibilité WCAG 2.0 A + AA.....	57
5.4.1	Identification des défauts de conformité.....	59
5.4.2	Remédiation pour le niveau A + AA	59
5.4.3	Résultat avant / après	61
6	Web App en Container – Préparation de l'environnement.....	63
6.1	Architecture générale	63
6.1.1	Architecture de la Web App Linux en Container.....	64
6.1.2	Architecture du pipeline pour le container Docker	64
6.2	Ajout de Docker à l'environnement de développement.....	65
6.2.1	Installation de Docker sur le poste de développement	65
6.2.2	Ajout du support ASP.NET et Docker dans Visual Studio.....	67
6.3	Création de l'application Web en container et publication dans GitHub	68
6.3.1	Création de l'application dans Visual Studio.....	68
6.3.2	Modifier le fichier Dockerfile	70
6.3.3	Test de l'application Docker en local	71
6.3.4	Publication de l'application dans GitHub	72
6.3.5	Ajout d'une page Web dans l'application	74
6.3.6	Mise à jour dans GitHub	77
6.4	Création de l'environnement Azure pour la Web App en Container	78
6.4.1	Création du référentiel de containers.....	78
6.4.2	Création des Web App for Containers	79
6.5	Création du pipeline DevOps	83
6.5.1	Création d'un nouveau Projet DevOps.....	83

6.5.2	Création du pipeline « Builds » DevOps (YAML)	84
6.5.3	Modification du fichier YAML.....	85
6.5.4	Configuration des Web App for Containers Azure.....	87
6.5.5	Création du pipeline « Release » du Container.....	89
6.6	Implémentation des tests d'accessibilité automatisés	92
6.7	Pipeline final	93
6.7.1	CI.....	93
6.7.2	CD	93

1 Introduction

Ce document décrit les étapes pas à pas nécessaires pour la mise en œuvre d'une implémentation de tests d'accessibilité dans un pipeline DevOps pour une application Web.

L'objectif étant de se rapprocher des pratiques de développement actuelles l'implémentation est réalisée sur deux pipelines différents :

- Une Web App classique
- Une Web App en Container Docker

Les tests de conformité permettent aussi de montrer comment corriger le code ASP.NET de l'application pour passer les tests et publier le site Web en production.

1.1 Prérequis

Les éléments suivants sont nécessaires pour la réalisation de ce prototype.

- PC Windows 10 ou Windows Serveur
- Visual Studio 2017 ou 2019
- Souscription Azure (<http://portal.azure.com>)
- Azure DevOps active (<http://dev.azure.com>)
- Profile GitHub existent (<https://github.com>)

2 Web App Classique - Préparation de l'environnement

Le premier prototype vise à construire et déployer en continu une application Web en mode PaaS, c'est-à-dire publiée par un service d'application Cloud, dans ce cas App Service Azure.

2.1 Architecture générale

Pour cette première application nous utiliserons la convention de noms suivante.

Nom	Composant
Application Web	
WebAppA11yClassic	Solution Visual Studio
WebAppA11yClassic	Référentiel GitHub
Ressources Azure	
WebAppA11yClassicRG	Resource group
WebAppA11yClassicStaging	App Service « Staging »
WebAppA11yClassicStagingPlan	App Service Plan « Staging »
WebAppA11yClassic	App Service « Production »
WebAppA11yClassicPlan	App Service Plan « Production »
Azure DevOps	
A11yDemo	Organisation DevOps
WebAppA11yClassic-ASP.NET-CI	Pipeline « Builds »
WebAppA11yClassic-ASP.NET-CD	Pipeline « Release »

2.1.1 Architecture Azure de la Web App Windows Classique

L'application est très simple, nous nous sommes basés sur une application Web ASP.NET Core proposée en standard par Visual Studio.

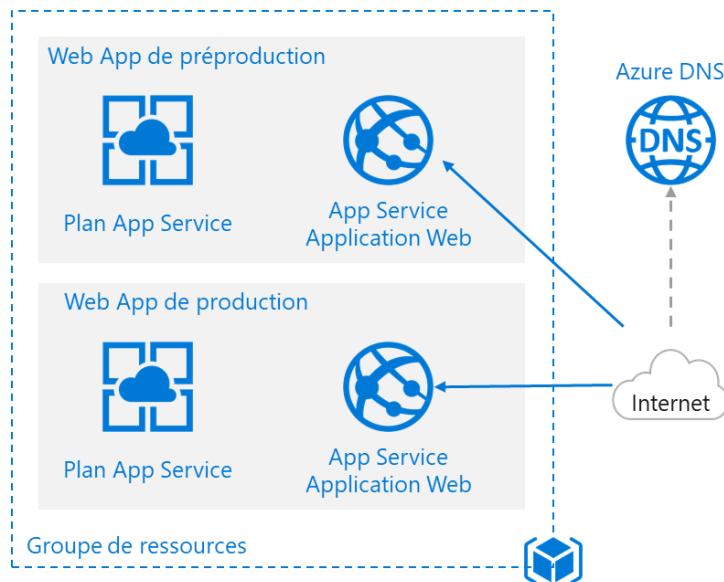


Figure 1 : Architecture de la Web App

Elle est stockée dans un référentiel GitHub. Elle est constituée de 3 pages : Home, Privacy, Accessibility.

2.1.2 Architecture du pipeline pour la Web App Classique

Ce pipeline doit s'activer automatiquement à chaque modification confirmée par un « commit » dans GitHub. Le pipeline débute par l'intégration continue (CI) et le déploiement continu (CD) pour construire et déposer l'application dans Azure.

La seconde partie est celle qui va contenir les tests d'accessibilité. Le pipeline de livraison « Release » publie l'application dans un premier Service Web App Azure de préproduction « Staging » puis exécute une série de tests d'accessibilité.

Si les tests sont validés alors le pipeline termine par la publication de l'application dans un second Service Web App de production.

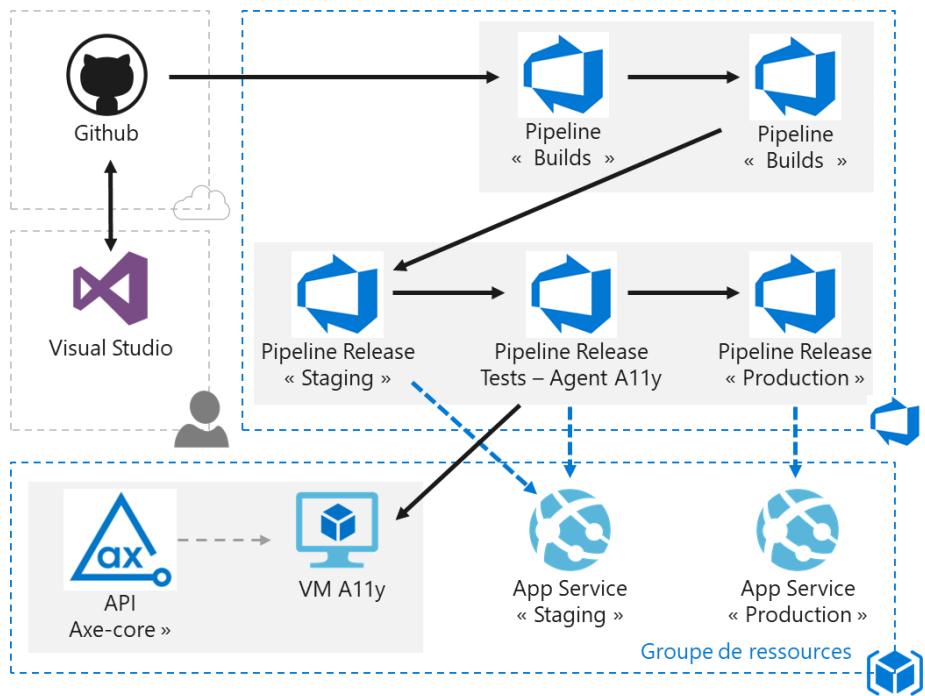


Figure 2 : pipeline de livraison continue pour la Web App

Les modifications effectuées dans Visual Studio peuvent être poussées dans GitHub et inversement.

2.2 Crédation de l'application Web et publication dans GitHub

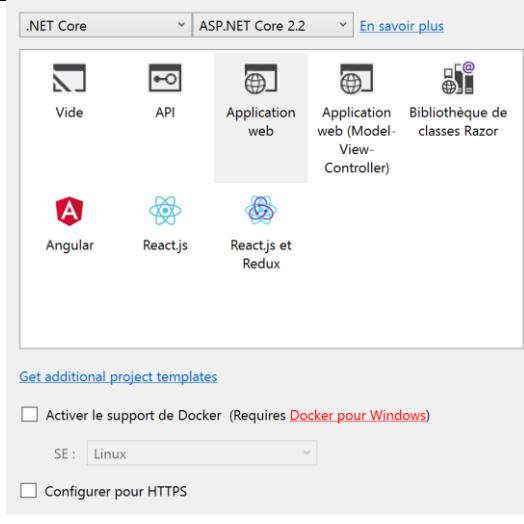
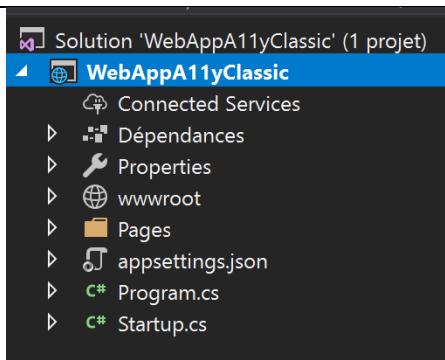
Pour cette préparation nous procéderons en 4 étapes :

- Crédation de l'application dans Visual Studio
- Publication dans GitHub
- Crédation d'une nouvelle page dans le projet de l'application
- Contrôle de mise à jour dans GitHub

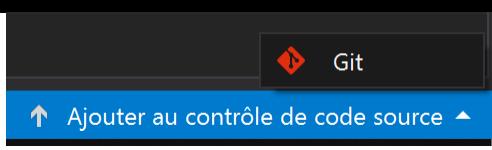
2.2.1 Crédation de l'application dans Visual Studio

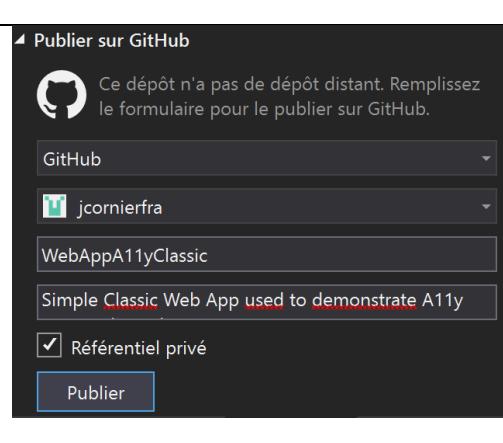
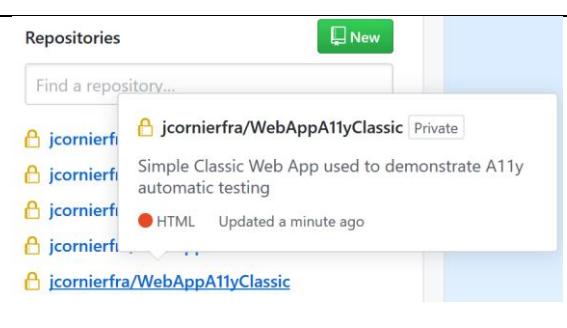
Illustration	Action
--------------	--------

	Lancer Visual Studio Créer un nouveau projet
	Sélectionner .NET Core
	Sélectionner Application web ASP.NET Core
<p>Nom : <input type="text" value="WebAppA11yClassic"/></p> <p>Emplacement : <input type="text" value="C:\Users\jcornier\source\repos\jcornierfra\"/></p> <p>Nom de solution : <input type="text" value="WebAppA11yClassic"/></p>	<p>Saisir le nom de l'application</p> <p>Ne nom de la solution (le même)</p> <p>Choisir le répertoire</p>
<input type="button" value="Parcourir..."/> <input checked="" type="checkbox"/> Créer un répertoire pour la solution <input type="checkbox"/> Créer un dépôt Git	Cocher Créer un répertoire pour la solution Ne pas cocher Créer un dépôt Git (nous utiliserons GitHub) Valider ces choix par OK

	<p>Vérifier que Application web est sélectionné</p> <p>Décocher HTTPS</p> <p>Ne pas cocher Docker</p> <p>OK</p>
	<p>Le projet apparaît dans Visual Studio / Explorateur de solutions</p>

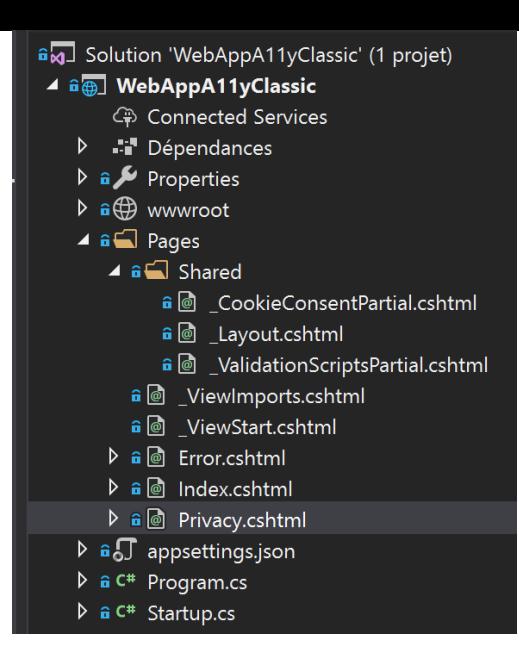
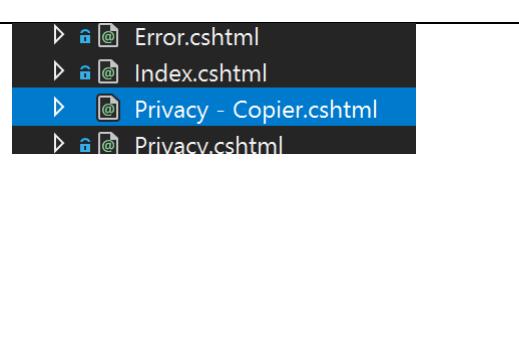
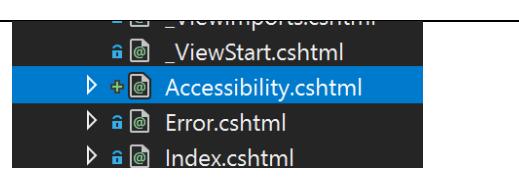
2.2.2 Publication de l'application dans GitHub

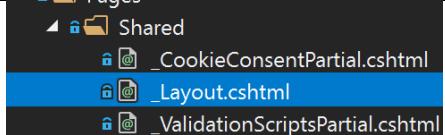
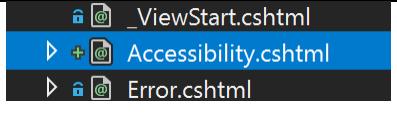
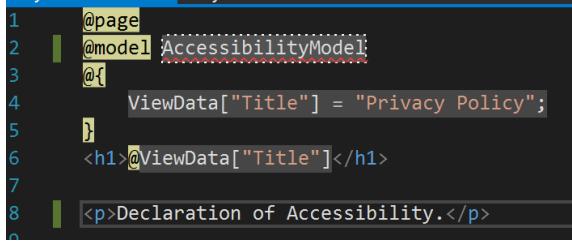
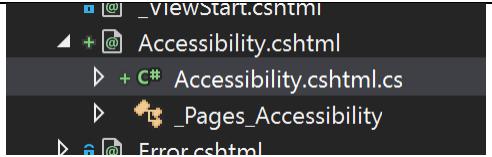
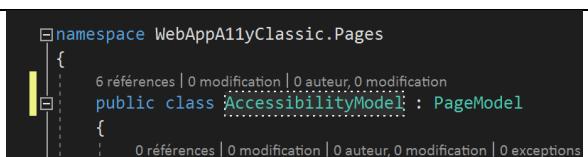
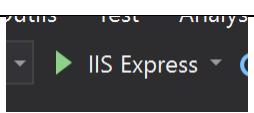
Illustration	Action
	<p>Cliquer en bas à droite sur Ajouter au contrôle de code source</p> <p>Cliquer sur Git</p>

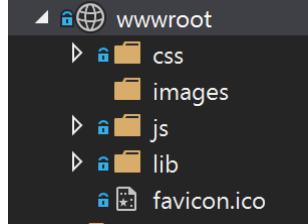
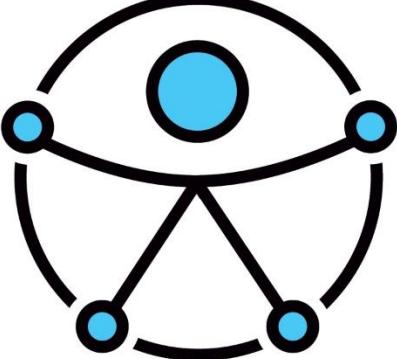
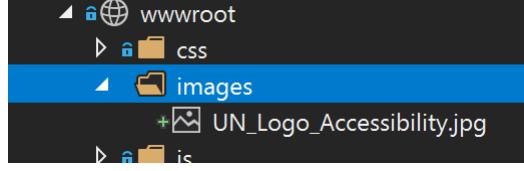
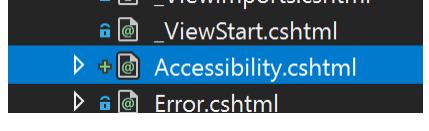
	<p>Cliquer sur Publier sur GitHub</p> <p>ATTENTION Azure DevOps est un autre dépôt. Nous voulons sélectionner GitHub</p>
	<p>Vérifier ou sélectionner le compte/profile GitHub</p> <p>Saisir le nom de la Web App</p> <p>Saisir une courte description (modifiable ultérieurement)</p> <p>Cocher Référentiel privé</p> <p>Publier</p>
	<p>Le dépôt a été créé</p>
	<p>Ouvrir https://github.com</p> <p>Cliquer sur le nom de la Web App</p>

<p>Simple Classic Web App used to demonstrate A11y automatic testing</p> <p>Manage topics</p> <p>2 commits 1 branch</p> <p>Branch: master New pull request</p> <p>jcornierfra Ajoutez des fichiers projet.</p> <ul style="list-style-type: none"> WebAppA11yClassic Ajoutez des fichiers projet. .gitattributes Ajoutez les fichiers .gitignore et .gitattributes. .gitignore Ajoutez les fichiers .gitignore et .gitattributes. WebAppA11yClassic.sln Ajoutez des fichiers projet. 	<p>Les fichiers sont présents</p>
---	--

2.2.3 Ajout d'une page Web dans l'application

Illustration	Action
	<p>Revenir dans Visual Studio, Explorateur de solutions</p> <p>Déployer Pages et Shared</p>
	<p>Cliquer bouton droit sur Privacy.cshtml</p> <p>Copier</p> <p>Cliquer bouton droit sur Pages</p> <p>Coller</p> <p>Le nouveau fichier Privacy-Copier apparaît</p>
	<p>Renommer en Accessibility.cshtml</p>

	Dans Shared, cliquer sur _Layout.cshtml Le contenu s'ouvre dans la fenêtre centrale
	Sélectionner et copier les 3 lignes correspondant à Privacy (Ctrl-C, Ctrl-V, Ctrl-V)
<pre>asp-page="/Index">Home asp-page="/Privacy">Privacy asp-page="/Accessibility">Accessibility</pre>	Renommer Privacy en Accessibility sur la nouvelle ligne Enregistrer (Ctrl-S)
	Sélectionner Accessibility.cshtml
	Remplacer Privacy par Accessibility Enregistrer
	Déplier Accessibility et sélectionner Accessibility.cshtml.cs
	Renommer PrivacyModel en AccessibilityModel Enregistrer
	Cliquer bouton droit sur la Solution Générer la solution
<pre>===== Génération : 1 a réussi, 0 a échoué, 0 mis à jour, 0 a été ignoré =====</pre>	La solution devrait se construire sans erreur
	Cliquer sur IIS Express

<h1>Welcome</h1> <p>Learn about building Web apps with ASP.NET Core.</p>	<p>La page Web s'ouvre</p> <p>Tester la navigation et la page d'accessibilité</p>
	<p>Stopper en cliquant sur le carré rouge</p>
	<p>Cliquer bouton droit sur wwwroot</p> <p>Ajouter</p> <p>Nouveau dossier</p> <p>Saisir le nom « images »</p> <p>Valider</p>
	<p>Enregistrer l'image ci-contre dans un fichier jpeg sous le nom</p> <p>UN_Logo_Accessibility.jpg</p>
	<p>Glisser et déposer le fichier image dans le dossier images</p>
	<p>Cliquer sur Accessibility.cshtml</p>

Ajouter le logo dans le code qui doit ressembler à celui-ci-dessous. Il manque le alt=« xxx » pour l'image, c'est volontaire pour les tests d'accessibilité.

```
@page
@model AccessibilityModel
 @{
     ViewData["Title"] = "Declaration of Accessibility";
 }
```

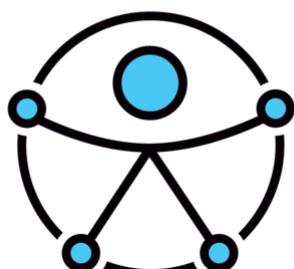
```

<div class="text-center">
    <h1>@ ViewData["Title"]</h1>

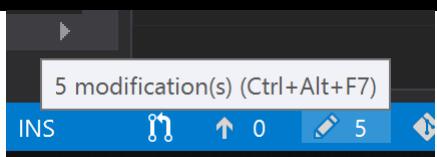
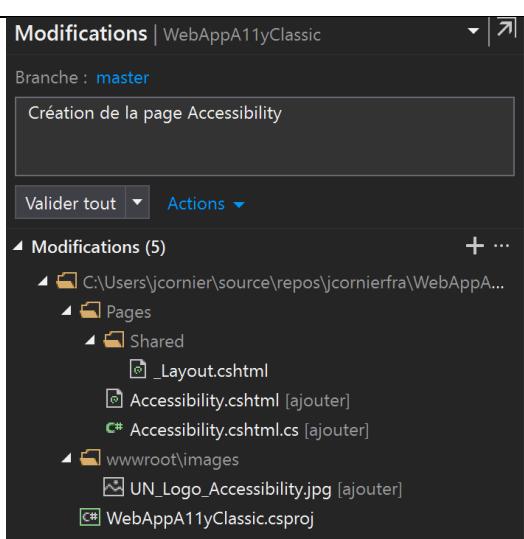
    <p>This Demo Web Site is targeted to reach the Standard WCAG 2.0 A + AA</p>

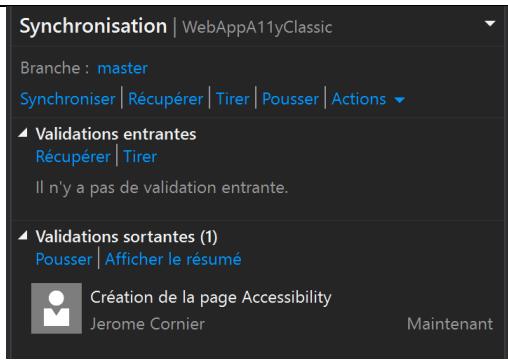
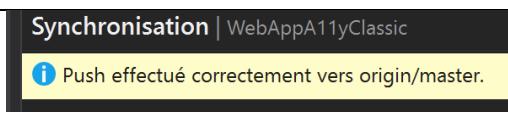
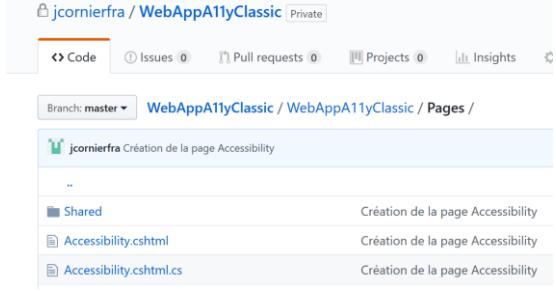
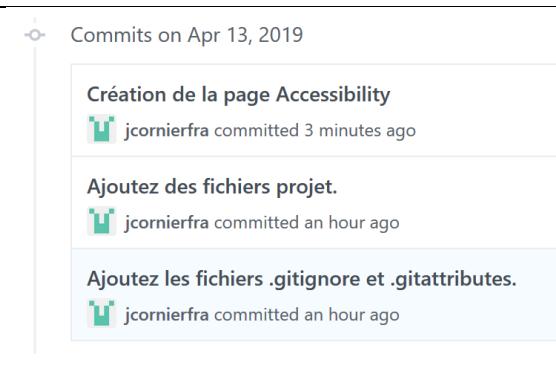
    <a href="https://www.un.org/fr/webaccessibility/logo.shtml">
        <img src "~/images/UN_Logo_Accessibility.jpg" width="300" />
    </a>
</div>

```

<h3>Declaration of Accessibility</h3> <p>This Demo Web Site is targeted to reach the Standard WCAG 2.0 A + AA</p> 	<p>Cliquer à nouveau sur IIS Express</p> <p>Vérifier le contenu de la nouvelle page</p>
---	---

2.2.4 Mise à jour dans GitHub

Illustration	Action
	<p>Cliquer sur le stylo en bas à droite (Autre solution : dans Team Explorer / Modifications)</p>
	<p>Entrer une description correspondant aux modifications effectuées Vérifier la liste des fichiers qui correspond à ceux modifiés précédemment Cliquer sur Valider tout</p>

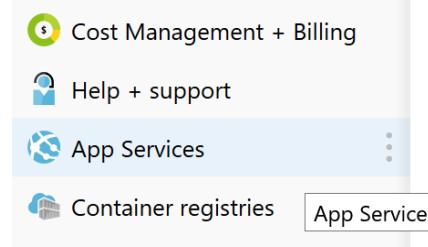
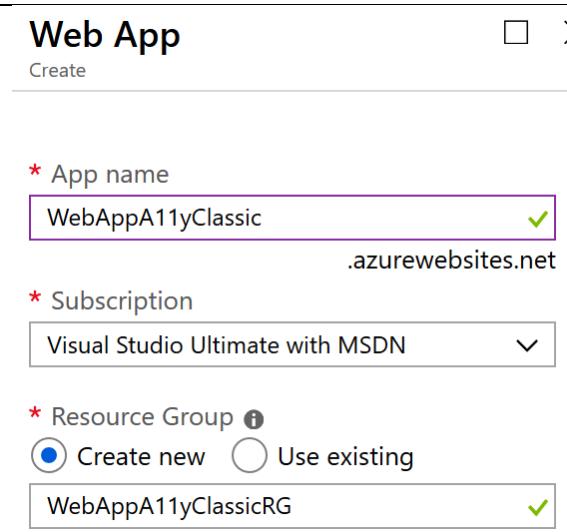
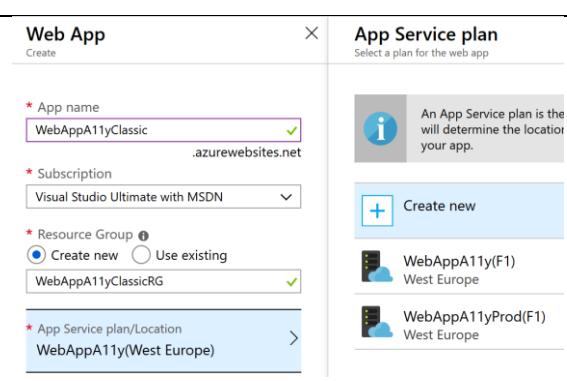
	Cliquer sur Synchroniser
	Cliquer sur Pousser
	La synchronisation a été effectuée
	<p>Ouvrir https://github.com</p> <p>Naviguer dans Pages</p> <p>Les nouveaux fichiers Accessibility doivent apparaître</p> <p>Idem pour le logo dans wwwroot/images</p>
	<p>Cliquer sur Commit</p> <p>Le dernier push apparaît, après la création du projet et l'initialisation du dépôt</p>

2.3 Création de l'environnement Azure

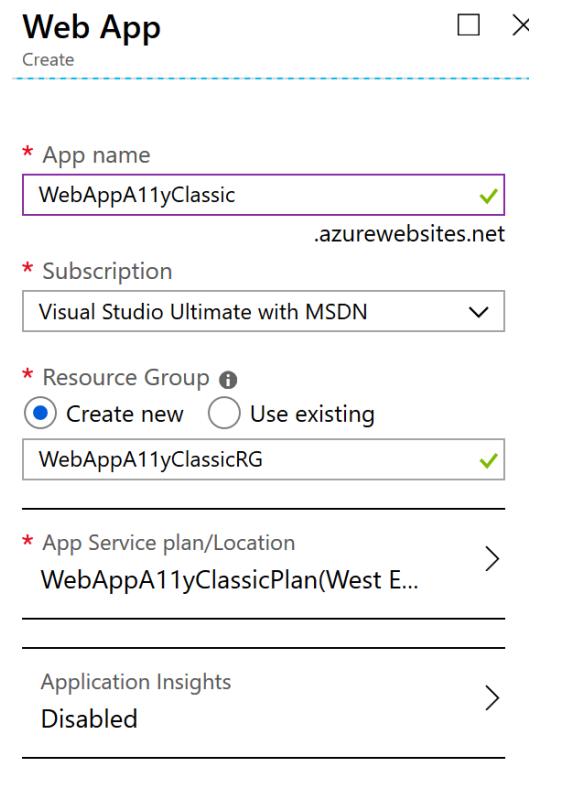
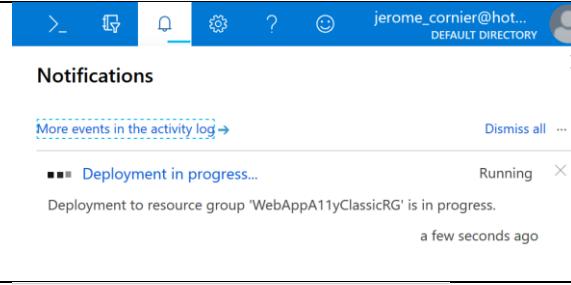
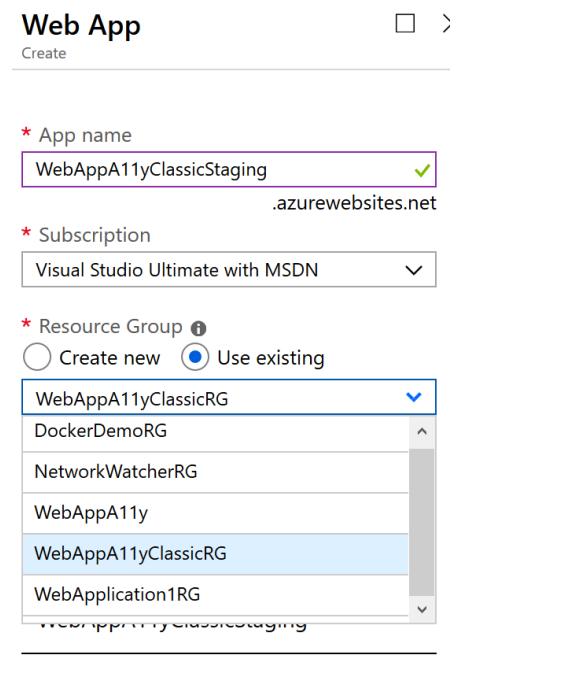
L'étape suivante consiste à préparer l'environnement Cloud pour recevoir l'application.

A partir du portail Azure (<http://portal.azure.com>) nous devons créer 2 Services d'application Web App avec leurs plans de services associés : la première Web App servira de préproduction

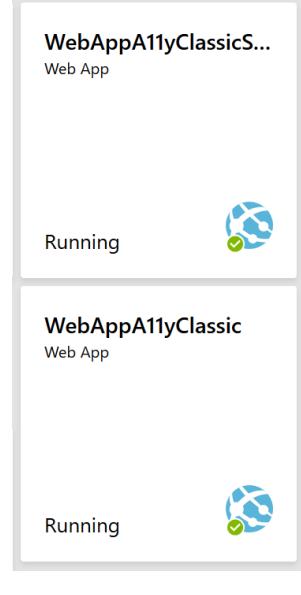
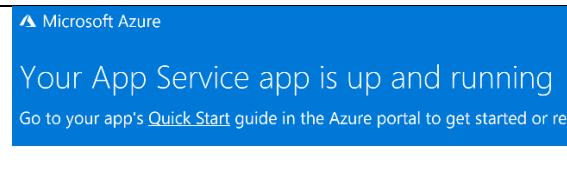
« Staging » pour tester l'accessibilité et la seconde sera déployée en production si les tests sont réussis.

Illustration	Action
	Créer une nouvelle ressource App Services
	<p>Add</p> <p>Saisir le nom de la Web App</p> <p>Choisir la souscription (si plusieurs)</p> <p>Créer en même temps le Resource Group en ajoutant « RG » à la fin du nom proposé</p>
	<p>Cliquer sur « App Service plan/Location</p> <p>Créer un nouveau plan</p>

	<p>Saisir le nom suivi de « Plan »</p> <p>Choisir l'emplacement (« West Europe » dans notre exemple)</p> <p>Cliquer sur Pricing tier pour changer le Plan par défaut en F1</p>
	<p>Cliquer sur Dev / Test</p> <p>Cliquer sur F1</p> <p>Apply</p>
	<p>Vérifier, puis OK</p>
	<p>Cliquer sur Application Insights</p>
<p>Application Insights site extensions</p> <p>Collect application monitoring data using App Insights</p> <p><input type="button" value="Enable"/> <input type="button" value="Disable"/> </p>	<p>Disable dans notre cas (pas utilisé dans le prototype actuel)</p> <p>Apply</p>

	<p>Vérifier, puis Créer</p>
	<p>Patienter quelques secondes</p>
	<p>Recommencer la création du second service pour le « Staging »</p> <p>Add</p> <p>Saisir le nom de la Web App avec « Staging » à la fin</p> <p>ATTENTION : Use existing Resource Group et sélectionner le RG créé précédemment</p>

	<p>Même opérations que précédemment pour le Service Plan et Application Insights</p> <p>Create</p>
	<p>Les deux App Services sont créés</p> <p>Clique pour chacun d'eux sur « Pin to Dashboard » pour les ajouter à la page d'accueil</p>

	<p>Les deux App Services sont actifs Cliquer sur l'un ou l'autre pour vérifier</p>
URL : https://webappa11yclassicstaging.azurewebsites.net App Service Plan : WebAppA11yClassicStagingPlan (F1: Free) FTP/deployment user... : No FTP/deployment user set FTP hostname : ftp://waws-prod-am2-175.ftp.azurewebsites.windows.net FTPS hostname : ftps://waws-prod-am2-175.ftp.azurewebsites.windows.net	<p>Cliquer sur l'URL qui apparaît</p>
	<p>Une page Web s'ouvre confirmant le bon fonctionnement Azure</p>

2.4 Crédit et configuration de l'organisation Azure DevOps

Le pipeline de livraison continue est ensuite créé en utilisant Azure DevOps (<http://dev.azure.com>).

L'opération nécessite plusieurs étapes :

- Crédit d'une organisation DevOps
- Crédit d'un projet
- Crédit d'un repository DevOps à partir de Github dans notre contexte
- Crédit du pipeline « Builds »
- Crédit du pipeline de livraison « Release »

Les étapes du pipeline sont les plus intéressantes pour ce prototypage, les précédentes sont juste des étapes de configuration basiques.

2.4.1 Crédation d'une nouvelle organisation DevOps

Illustration	Action
 New organization  Organization settings	Nouvelle organisation
 Azure DevOps jerome_cornier@hotmail.com Get started with Azure DevOps <p>Choosing Continue means that you agree to our Terms of Service, Privacy Statement, and Code of Conduct.</p> <p style="text-align: center;">Continue</p>	Continue
 Azure DevOps Almost done... Name your Azure DevOps organization <input type="text" value="dev.azure.com/ A11yDemo"/> We'll host your projects in * <input type="text" value="West Europe"/> Entrez les caractères que vous voyez Nouveau Fichier audio  <p style="text-align: center;">Continue</p>	Saisir le nom de l'organisation et vérifier la localisation Continue

2.4.2 Crédation d'un nouveau Projet DevOps

Illustration	Action
--------------	--------

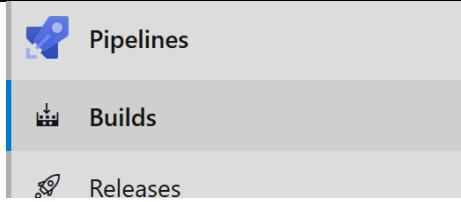
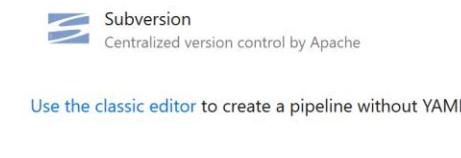
<p>Create a project to get started</p> <p>Project name * WebAppA11yClassic</p> <p>Description Classic Web App to demo automatic accessibility testing</p> <p>Visibility <input checked="" type="radio"/> Public Private <small>Only people you give access to will be able to view this project.</small> </p> <p><input type="button" value="Advanced"/></p> <p><input type="button" value="Create project"/></p>	<p>Entrer le nom de la Web App</p> <p>Entrer une description</p> <p>Choisir Private</p> <p>Create project</p>
--	---

2.4.3 Crédit du pipeline « Builds »

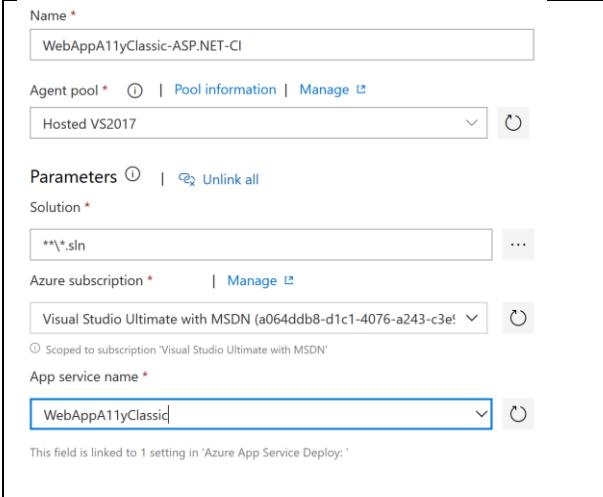
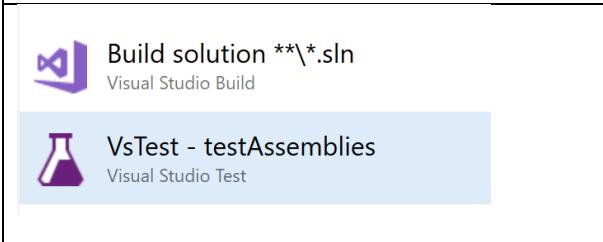
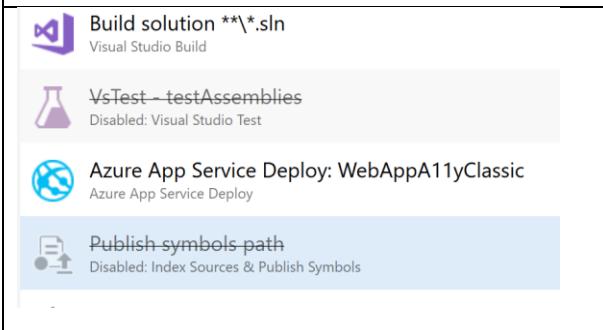
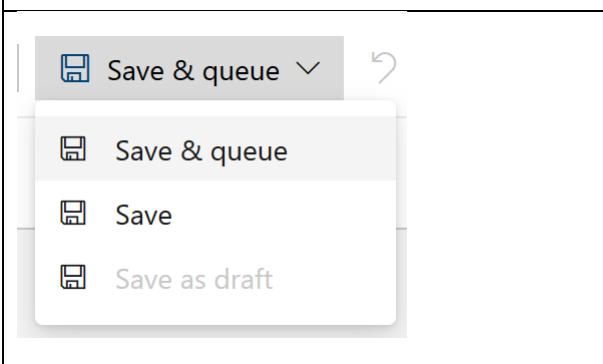
« Builds » signifie Continuous Integration et Continuous Deployment. Cette partie du pipeline permet de construire et envoyer la solution dans Azure mais le Service d'Application n'est pas mis à jour.

Il existe deux manières de créer le pipeline « Builds » : soit en utilisant les briques de DevOps, soit en fournissant un fichier YAML (Yet Another Markup Language) qui correspond à un script d'installation.

Pour cette Web App nous avons choisi les briques DevOps qui illustrent mieux le fonctionnement. Nous l'avons simplifié en ôtant certains tests qui ne s'appliquent pas à notre application.

Illustration	Action
	Cliquer sur Pipelines
<p>No build pipelines were found</p> <p>Automate your build in a few easy steps with a new pipeline.</p> <p><input type="button" value="New pipeline"/></p>	New pipeline
	Cliquez sur Use the classic editor

Select a source	Cliquer sur GitHub
	
Connection name * <input type="text" value="https://github.com/jcornierfra/WebAppA11yClassic"/> Authorize using OAuth Or Authorize with a GitHub token	Copier l'URL du dépôt GitHub Coller dans le champ Connexion name Cliquer sur Authorize using OAuth Saisir les credentials si besoin
Repository * Manage on GitHub <input type="text" value="jcornierfra/WebAppA11yClassic"/> Default branch for manual and scheduled builds * <input type="text" value="master"/> Continue	Sélectionner le dépôt de la Web App Branch master par défaut Continue
 Android Build, test, sign, and align an Android APK.  ASP.NET Build and test an ASP.NET web application.  Azure Web App for ASP.NET Build, package, test, and deploy an ASP.NET Azure Web App.  Docker container Build a Docker image and push it to a container registry.	Sélectionner le template Azure Web App for ASP.NET Apply (le bouton Apply apparaît à droite de la ligne où se trouve la souris)
Name * <input type="text" value="WebAppA11yClassic-ASP.NET-CI"/> Agent pool * (i) Pool information Manage <input type="text" value="Hosted VS2017"/> Parameters (i) Unlink all Solution * <input type="text" value="***.sln"/> Azure subscription * Manage <input style="width: 150px; margin-right: 10px;" type="text" value="Visual Studio Ultimate with MSDN (a064ddb8-c...)"/> Authorize (i)  <small>(i) Click Authorize to configure an Azure service connection</small>	Mettre à jour le nom Sélectionner la souscription Azure Cliquer sur Authorize Saisir les credentials dans la fenêtre qui s'affiche

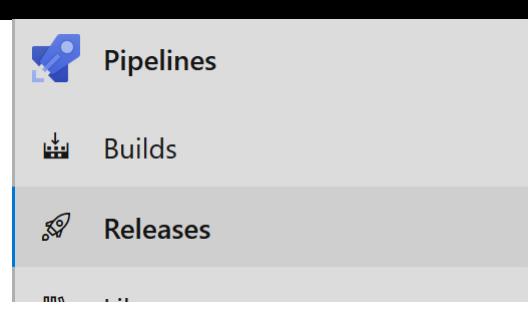
 <p>Name * <input type="text" value="WebAppA11yClassic-ASP.NET-CI"/></p> <p>Agent pool * <input type="text" value="Hosted VS2017"/> Pool information Manage</p> <p>Parameters <input type="text" value="Unlink all"/> Manage</p> <p>Solution * <input type="text" value="***.sln"/> ...</p> <p>Azure subscription * <input style="background-color: #f0f0f0; border: 1px solid #ccc; width: 200px; height: 20px; vertical-align: middle;" type="text" value="Visual Studio Ultimate with MSDN (a064ddb8-d1c1-4076-a243-c3e1...)"/> Manage</p> <p>App service name * <input type="text" value="WebAppA11yClassic"/> Manage</p> <p>This field is linked to 1 setting in 'Azure App Service Deploy.'</p>	<p>Rafraîchir si besoin avec les boutons à droite des champs</p> <p>Sélectionner la Web App créée précédemment</p>
 <p>Build solution ***.sln Visual Studio Build</p> <p>VsTest - testAssemblies Visual Studio Test</p>	<p>Cliquer sur VsTest – testAssemblies</p>
<p>Control Options ^</p> <p><input type="checkbox"/> Enabled</p> <p><input type="checkbox"/> Continue on error</p> <p>—</p>	<p>Parcourir la liste des paramètres</p> <p>Déplier Control Options</p> <p>Décocher Enabled</p> <p>Ce test n'est pas compatible avec GitHub</p>
 <p>Build solution ***.sln Visual Studio Build</p> <p>VsTest - testAssemblies Disabled: Visual Studio Test</p> <p>Azure App Service Deploy: WebAppA11yClassic Azure App Service Deploy</p> <p>Publish symbols path Disabled: Index Sources & Publish Symbols</p>	<p>Même opération pour Publish symbols path</p>
 <p>Save & queue</p> <p>Save & queue</p> <p>Save</p> <p>Save as draft</p>	<p>Save & queue</p>

 Build #20190413.1 has been queued.	Cliquer sur le numéro de Build																																	
Agent job 1 Job Started: 13/04/2019 à 16:34:49 Pool: Hosted VS2017 · Agent: Hosted Agent ... 5m 37s <table border="1" data-bbox="208 451 743 968"> <tr><td>✓ Initialize Agent</td><td>succeeded</td><td>1s</td></tr> <tr><td>✓ Prepare job</td><td>succeeded</td><td><1s</td></tr> <tr><td>✓ Initialize job</td><td>succeeded</td><td>5s</td></tr> <tr><td>✓ Checkout</td><td>succeeded</td><td>12s</td></tr> <tr><td>✓ Use NuGet 4.4.1</td><td>succeeded</td><td>1s</td></tr> <tr><td>✓ NuGet restore</td><td>succeeded</td><td>3m 18s</td></tr> <tr><td>✓ Build solution ***.sln</td><td>succeeded</td><td>1m 26s</td></tr> <tr><td>✓ Azure App Service Deplo...</td><td>succeeded</td><td>31s</td></tr> <tr><td>✓ Publish Artifact: drop</td><td>succeeded</td><td>1s</td></tr> <tr><td>✓ Post-job: Checkout</td><td>succeeded</td><td><1s</td></tr> <tr><td>✓ Finalize Job</td><td>succeeded</td><td><1s</td></tr> </table>	✓ Initialize Agent	succeeded	1s	✓ Prepare job	succeeded	<1s	✓ Initialize job	succeeded	5s	✓ Checkout	succeeded	12s	✓ Use NuGet 4.4.1	succeeded	1s	✓ NuGet restore	succeeded	3m 18s	✓ Build solution ***.sln	succeeded	1m 26s	✓ Azure App Service Deplo...	succeeded	31s	✓ Publish Artifact: drop	succeeded	1s	✓ Post-job: Checkout	succeeded	<1s	✓ Finalize Job	succeeded	<1s	Surveiller les tâches Le pipeline complet dure plusieurs minutes
✓ Initialize Agent	succeeded	1s																																
✓ Prepare job	succeeded	<1s																																
✓ Initialize job	succeeded	5s																																
✓ Checkout	succeeded	12s																																
✓ Use NuGet 4.4.1	succeeded	1s																																
✓ NuGet restore	succeeded	3m 18s																																
✓ Build solution ***.sln	succeeded	1m 26s																																
✓ Azure App Service Deplo...	succeeded	31s																																
✓ Publish Artifact: drop	succeeded	1s																																
✓ Post-job: Checkout	succeeded	<1s																																
✓ Finalize Job	succeeded	<1s																																

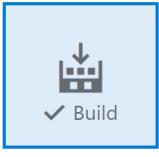
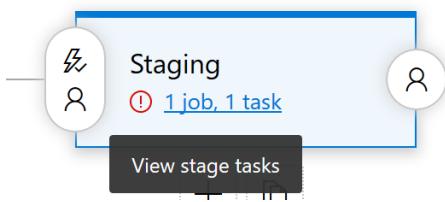
2.4.4 Crédit du pipeline « Release »

Le pipeline de livraison peut être paramétré pour démarrer dès que le pipeline « Builds » a terminé son déploiement, c'est ce que nous avons configuré.

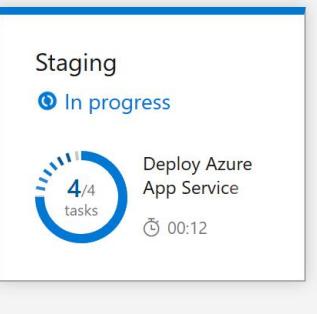
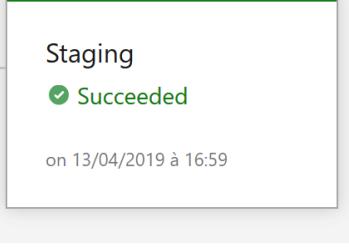
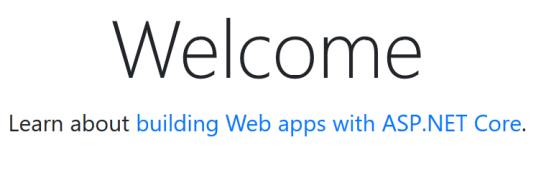
L'étape suivante consiste à mettre en place le pipeline de déploiement de la Web App.

Illustration	Action
	Sélectionner Releases
No release pipelines found Automate your release process in a few easy steps with a new pipeline <a data-bbox="409 1837 530 1866" href="#">New pipeline	New pipeline

<p>Featured</p> <p> Azure App Service deployment Deploy your application to Azure App Service. Choose from Web App on Windows, Linux, containers, Function Apps, or WebJobs.</p>	<p>Choisir Azure App Service deployment</p> <p>Apply</p>
<p>Stage</p> <p>Staging</p> <p> Properties ^ Name and owners of the stage</p> <p>Stage name <input type="text" value="Staging"/></p> <p>Stage owner  Jerome Cornier</p>	<p>Renommer en « Staging »</p>
<p>Artifacts  Add</p> <p> Add an artifact</p> <p> Schedule not set</p>	<p>Cliquer sur Add an artifact</p>

<p>Source type</p>  <p>Build</p> <p>Azure Repos ...</p> <p>4 more artifact types ▾</p> <p>Project * ⓘ</p> <p>WebAppA11yClassic</p> <p>Source (build pipeline) * ⓘ</p> <p>WebAppA11yClassic-ASP.NET-CI</p> <p>Default version * ⓘ</p> <p>Latest</p> <p>Source alias * ⓘ</p> <p>_WebAppA11yClassic-ASP.NET-CI</p> <p><small>(i) The artifacts published by each version will be successful build of WebAppA11yClassic-AS</small></p> <p>Add</p>	<p>Choisir la Source (build pipeline) qui est le pipeline précédemment créé</p> <p>Les autres champs sont automatiquement renseignés</p> <p>Add</p>
<p>Artifacts + Add</p>  <p>_WebAppA11yClassic-ASP.NET-CI</p>	<p>Cliquer sur l'icône de l'éclair dans le coin de la boîte de l'Artifact</p>
<p>Continuous deployment trigger</p> <p>Build: _WebAppA11yClassic-ASP.NET-CI</p> <p><input checked="" type="checkbox"/> Enabled</p> <p>Creates a release every time a new build is available.</p> <p>Build branch filters ⓘ</p>	<p>Cliquer sur l'interrupteur pour activer le déclenchement automatique du pipeline Release</p>
 <p>Staging</p> <p>(i) 1 job, 1 task</p> <p>View stage tasks</p>	<p>Cliquer sur 1 job, 1 task</p>

	<p>Sélectionner la souscription Azure</p> <p>Sélectionner Web App on Windows</p> <p>Sélectionner la Web App "Staging »</p>
	<p>Positionner la souris sur New release pipeline</p> <p>Cliquer sur le stylo</p>
	<p>Changer le nom du pipeline</p>
	<p>Save</p> <p>OK</p>
	<p>Cliquer sur All pipelines</p>
	<p>Cliquer sur le nom du pipeline</p>
	<p>Create release</p> <p>Create</p>
	<p>Cliquer sur le nom de la Release en cours</p>

	Patienter
	Le déploiement devrait être réussi
	Ouvrir le portail Azure Sélectionner la Web App Staging
	Cliquer sur son URL
	La page du site publié dans Azure doit apparaître

3 Création de l'Agent Azure DevOps spécifique A11y

Nous avons retenu l'extension A11y Accessibility Testing développée par Drew Lewis [108] qui présente les avantages suivants :

- Cette extension s'appuie sur l'API libre axe-core développée et supportée par Deque University [109]
- L'API axe-core est sous licence MPL 2.0
- Elle a pour philosophie « zéro faux positifs »
- C'est une extension configurable pour un pipeline Azure DevOps
- Le principe est l'analyse du site en ligne, l'extension fonctionne quel que soit le code source et le type d'hébergement
- Standards supportés : WCAG Level A, WCAG Level AA, Section 508

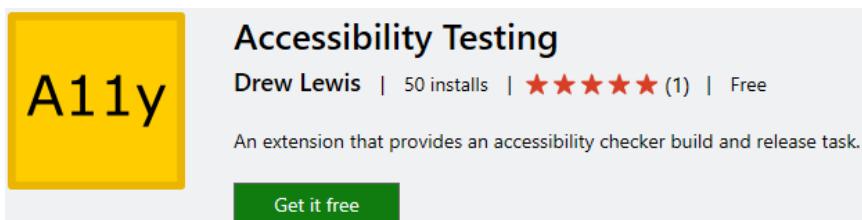


Figure 3 : A11y Extension Accessibility Testing, Drew Lewis

3.1 Architecture Générale

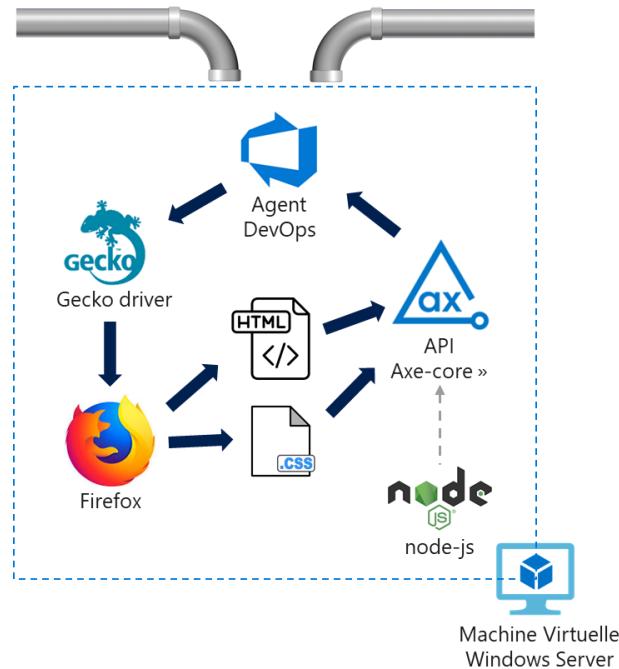
3.1.1 Architecture Azure

Nom	Composant
Ressources Azure	
VMAgentA11yRG	Resource group
VMAgentA11y	Virtual machine
VMAgentA11y_DataDisk_0	Disk 0 – nom automatique
VMAgentA11y...	Disk 1 – nom automatique
Vmagenta11y...	Network interface – nom automatique

VMAgentA11y-ip	Public IP address – nom automatique
VMAgentA11y-nsg	Network security group – nom automatique
VMAgentA11yRG-vnet	Virtual network – nom automatique

3.1.2 Architecture de la machine virtuelle de l'Agent A11y

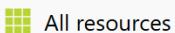
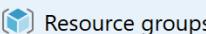
La solution est construite dans une Machine Virtuelle intégrée au pipeline CD.



3.2 Crédit de la VM Azure pour l'Agent DevOps

Notre solution de tests d'accessibilité requiert plusieurs dépendances qui s'installent sur Windows. Nous avons donc choisi une machine virtuelle Azure préparée avec Windows Server 2016 Datacenter.

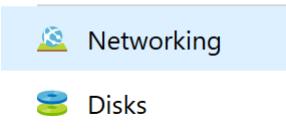
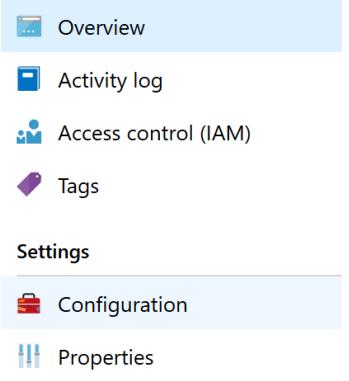
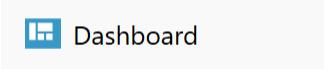
Le modèle de machine n'a pas besoin de puissance pour notre Web App, nous avons sélectionné le minimum auquel nous avons ajouté un disque SSD de 200 Go pour stocker les logiciels.

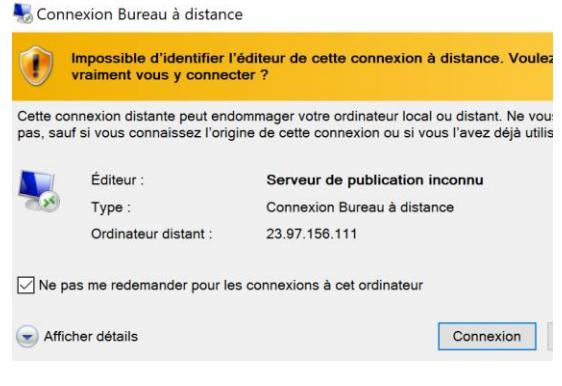
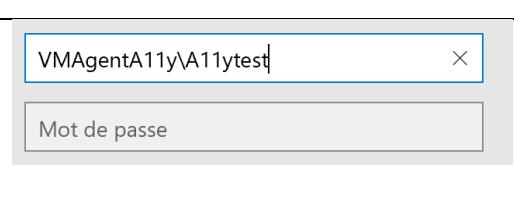
Illustration	Action
  	<p>Dans le portail Azure</p> <p>http://portal.azure.com</p>

	<p>Cliquer sur Resource groups</p> <p>Add</p>
<p>Visual Studio Ultimate with MSDN</p> <p>VMAgentA11yRG</p> <p>West Europe</p>	<p>Sélectionner la souscription et la localisation</p> <p>Entrer le nom du resource group</p> <p>Review + Create</p> <p>Create</p>
<p> Resource group created Creating resource group 'VMAgentA11yRG' in subscription 'Ultimate with MSDN' succeeded.</p> <p>Go to resource group Pin to dashboard</p>	Attendre la confirmation
<p> Azure Cosmos DB</p> <p> Virtual machines</p> <p> Load balancers</p>	<p>Cliquer sur Virtual machines</p> <p>Add</p>
<p>Visual Studio Ultimate with MSDN</p> <p>VMAgentA11yRG</p> <p>Create new</p> <p>VMAgentA11y</p> <p>West Europe</p> <p>No infrastructure redundancy required</p> <p>Windows Server 2016 Datacenter</p> <p>Browse all images</p>	<p>Choisir la souscription</p> <p>Entrer le nom de la machine</p> <p>Choisir Windows Server 2016 Datacenter</p>
<p>Standard DS1 v2 1 vcpu, 3.5 GB memory Change size</p>	<p>Garder le DS1 v2 par défaut</p> <p>Environ 42€ / mois</p>

	Saisir le nom de l'utilisateur Admin Saisir 2 fois le mot de passe Conserver ces credentials pour les prochaines étapes
	Ajouter le port RDP 3389
	Next : Disks pour ajouter un disque persistent
	Standard SSD est suffisant
	Cliquer sur Create and attach a new disk
	Standard SSD, 200Gb OK
	Review + create
	Contrôler Create
	Patienter

<p> Deployment succeeded</p> <p>Deployment 'CreateVm-MicrosoftWindowsServer.W201-20190413175312' to resource group 'VMAgentA11y' successful.</p> <p>Go to resource Pin to dashboard</p>	<p>Pin to dashboard</p>																		
<table border="1"> <thead> <tr> <th>RESOURCE</th><th>TYPE</th></tr> </thead> <tbody> <tr> <td> shutdown-computevm-VMAgentA11y</td><td>Microsoft.DevTestLab/schedules</td></tr> <tr> <td> VMAgentA11y</td><td>Microsoft.Compute/virtualMachines</td></tr> <tr> <td> vmagenta11y491</td><td>Microsoft.Network/networkInterfaces</td></tr> <tr> <td> VMAgentA11yRG-vnet</td><td>Microsoft.Network/virtualNetworks</td></tr> <tr> <td> VMAgentA11y-nsg</td><td>Microsoft.Network/networkSecurityGroups</td></tr> <tr> <td> VMAgentA11y-ip</td><td>Microsoft.Network/publicIPAddresses</td></tr> <tr> <td> vmagenta11yrgdiag</td><td>Microsoft.Storage/storageAccounts</td></tr> <tr> <td> VMAgentA11y_DataDisk_0</td><td>Microsoft.Compute/disks</td></tr> </tbody> </table>	RESOURCE	TYPE	 shutdown-computevm-VMAgentA11y	Microsoft.DevTestLab/schedules	 VMAgentA11y	Microsoft.Compute/virtualMachines	 vmagenta11y491	Microsoft.Network/networkInterfaces	 VMAgentA11yRG-vnet	Microsoft.Network/virtualNetworks	 VMAgentA11y-nsg	Microsoft.Network/networkSecurityGroups	 VMAgentA11y-ip	Microsoft.Network/publicIPAddresses	 vmagenta11yrgdiag	Microsoft.Storage/storageAccounts	 VMAgentA11y_DataDisk_0	Microsoft.Compute/disks	<p>Vérifier les ressources créées</p>
RESOURCE	TYPE																		
 shutdown-computevm-VMAgentA11y	Microsoft.DevTestLab/schedules																		
 VMAgentA11y	Microsoft.Compute/virtualMachines																		
 vmagenta11y491	Microsoft.Network/networkInterfaces																		
 VMAgentA11yRG-vnet	Microsoft.Network/virtualNetworks																		
 VMAgentA11y-nsg	Microsoft.Network/networkSecurityGroups																		
 VMAgentA11y-ip	Microsoft.Network/publicIPAddresses																		
 vmagenta11yrgdiag	Microsoft.Storage/storageAccounts																		
 VMAgentA11y_DataDisk_0	Microsoft.Compute/disks																		
<p> Your deployment is complete</p> <p>Go to resource</p> <div data-bbox="244 968 323 1051">  </div> <p>Deployment name: CreateVm-MicrosoftWindowsServer.W201-20190413175312 Subscription: Visual Studio Ultimate with Microsoft 365 Dev/Collab Resource group: VMAgentA11yRG</p>	<p>Go to resource</p>																		
<p>Operations</p> <p> Auto-shutdown</p> <p> Backup</p>	<p>Cliquer sur Auto-shutdown</p>																		
<p> Save  Discard  Fee</p> <p>Enabled <input checked="" type="button"/> On <input type="button"/> Off</p> <p>Scheduled shutdown <input type="text" value="10:00:00 PM"/></p> <p>Time zone <input type="text" value="(UTC) Coordinated Universal Time"/></p> <p>Send notification before auto-shutdown <input checked="" type="button"/> Yes <input type="button"/> No</p> <p>Webhook URL </p> <p>Email address </p> <p>fff@microsoft.com</p>	<p>On</p> <p>Choisir une heure d'arrêt automatique</p> <p>Envoyer (ou pas) une notification</p> <p>Saisir l'adresse email</p> <p>Save</p>																		

<p>Settings</p>  <p>Networking</p> <p>Disks</p>	<p>Cliquer sur Networking</p> <p>Cliquer sur Public IP</p>
<p>VMAgentA11y-ip Public IP address</p> <p>Search (Ctrl+ /)</p>  <p>Overview</p> <p>Activity log</p> <p>Access control (IAM)</p> <p>Tags</p> <p>Settings</p> <p>Configuration</p> <p>Properties</p>	<p>Cliquer sur Configuration</p>
 The associated virtual machine is not assigned to a static IP address. <p>Assignment</p> <p><input type="radio"/> Dynamic <input checked="" type="radio"/> Static</p>	<p>Sélectionner Static</p> <p>Save</p>
<p>Dashboard</p> 	<p>Cliquer sur Dashboard</p>
<p>VMAgentA11y</p> <p>Running</p> 	<p>Cliquer sur le VMAgentA11y</p>
<p>Connect</p> 	<p>Cliquer sur Connect</p>
<p>* Port number</p> <p>3389</p> <p>Download RDP File</p> 	<p>Cliquer sur Download RDP File</p> <p>Enregistrer le fichier sur le Bureau</p>

	<p>Double cliquer sur le fichier RDP</p> <p>Cocher la case Ne pas me redemander...</p> <p>Connexion</p>
<p>Autres choix</p>	<p>Cliquer sur Autres choix</p>
<p> Utiliser un autre compte</p>	<p>Utiliser un autre compte</p>
	<p>Entrer les credentials fournies précédemment lors de la création de la VM</p> <p>OK</p>
<p>Voulez-vous vous connecter malgré ces erreurs de certificat ?</p> <p><input checked="" type="checkbox"/> Ne pas me redemander pour les connexions à cet ordinateur</p> <p>Afficher le certificat... Oui</p>	<p>Cocher la case</p> <p>Oui</p>
	<p>Patienter le temps du chargement complet de la VM</p>

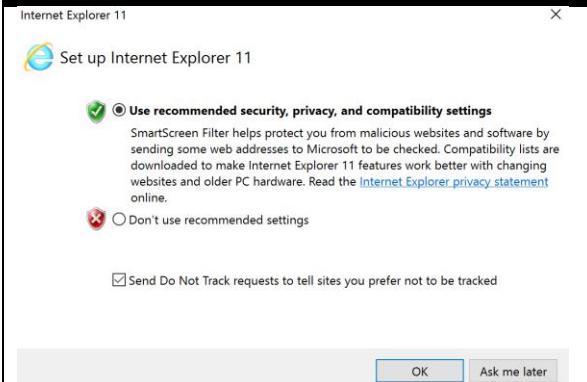
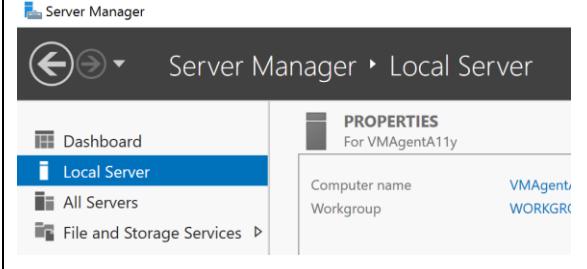
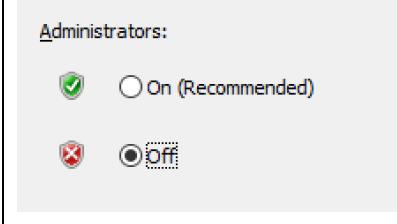
3.3 Installation des dépendances logicielles

Les dépendances logicielles sont :

- L'Agent DevOps pour Windows qu'il faut associer au compte utilisateur pour DevOps [110]. Le lien permettant de télécharger l'agent est disponible dans les paramètres du projet dans DevOps ainsi que les commandes d'installation
- Navigateur Firefox pour Windows

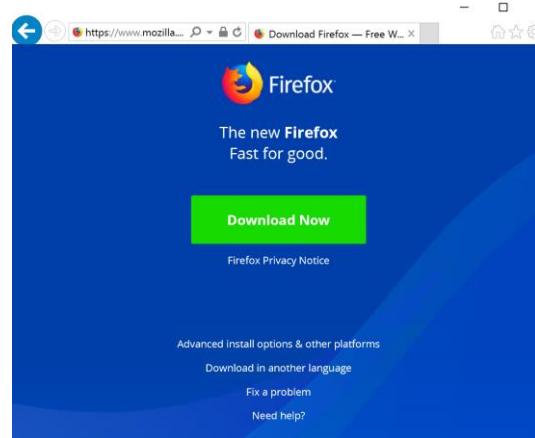
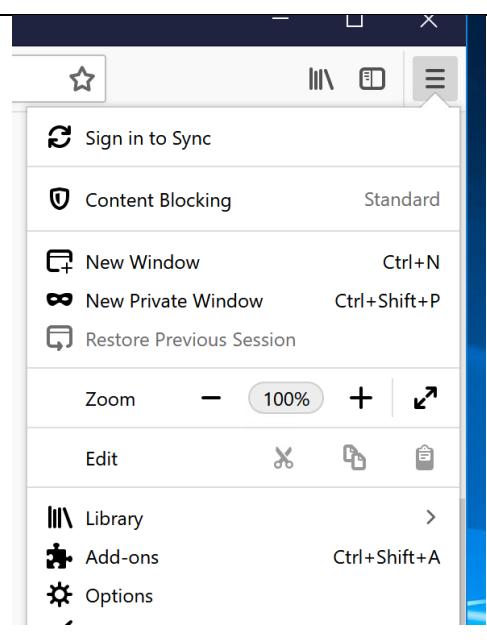
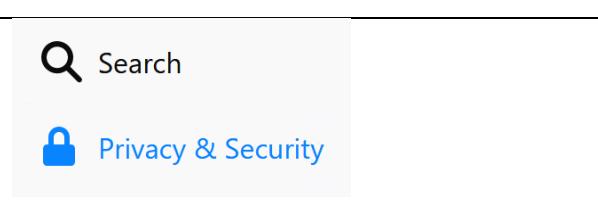
- Gecko driver
- Node.js qui est inclus par défaut dans la VM Azure

3.3.1 Configuration Internet

Illustration	Action
	<p>Lancer Internet Explorer</p> <p>Valider la recommandation</p> <p>Fermer Internet Explorer</p>
	<p>Ouvrir Server Manager</p> <p>Cliquer sur Local Server</p>
Feedback & Diagnostics IE Enhanced Security Configuration On	Cliquer sur On
	<p>Sélectionner Off</p> <p>OK</p>

3.3.2 Installation de Firefox

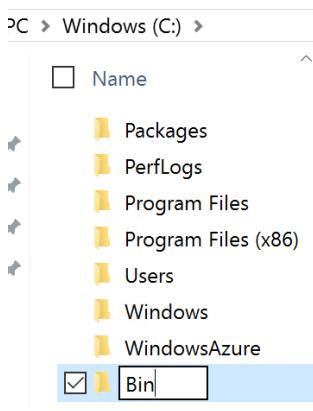
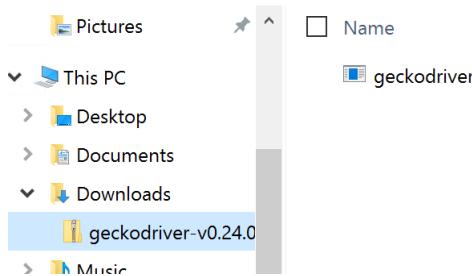
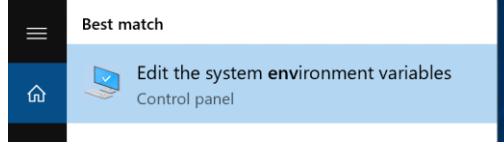
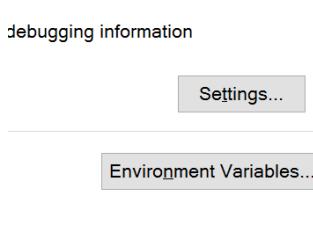
Illustration	Action
--------------	--------

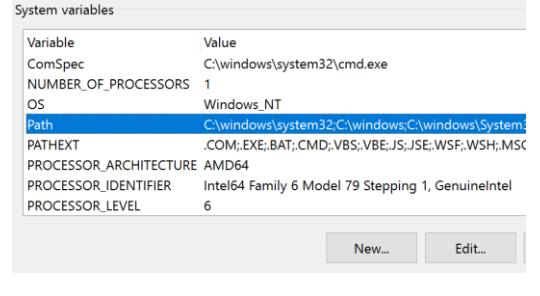
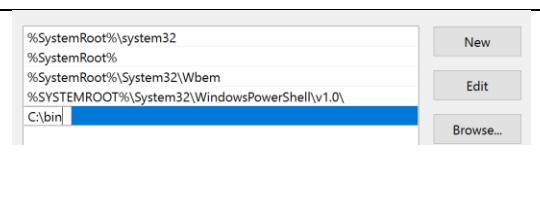
	<p>Ouvrir Internet Explorer</p> <p>Taper Firefox</p> <p>Ouvrir la page Web de Firefox</p> <p>Cliquer sur Download Now</p> <p>Run</p>
	<p>Patienter</p>
	<p>Firefox d'ouvre</p> <p>Cliquer sur le menu (3 barres en haut à droite)</p> <p>Cliquer sur Options</p>
	<p>Cliquer sur Privacy & Security</p>

Cookies and Site Data <p>Your stored cookies, site data and cache are currently using space. Learn more</p> <p><input checked="" type="checkbox"/> Delete cookies and site data when Firefox is closed</p>	Cacher / Supprimer les cookies ...
History <p>Firefox will Never remember history</p> <p>Firefox will use the same settings as private browsing, and will not remember any history as you browse the Web.</p> <p>Address Bar</p> <p>When using the address bar, s</p> <p><input type="checkbox"/> Firefox must restart to enable this feature.</p> <p>Restart Firefox Cancel</p>	Sélectionner Never remember history Restart Firefox now
Default Browser <p><input checked="" type="checkbox"/> Firefox is not currently set as your default browser. Would you like to make it the default?</p> <p><input checked="" type="checkbox"/> Always perform this check when starting Firefox.</p> <p>Use Firefox as my default browser</p>	Cliquer sur Use Firefox as my default browser
 <p>Choose an app</p> <p>Video</p> <p>Firefox</p> <p>Web browser</p> <p>Internet Explorer</p> <p>Web browser</p> <p>Internet Explorer</p>	La fenêtre des paramètres s'ouvre Sélectionner Firefox Fermer la fenêtre Fermer Firefox et IE

3.3.3 Installation du pilote Gecko

Illustration	Action
	Lancer Firefox Taper l'URL https://github.com/mozilla/geckodriver/releases

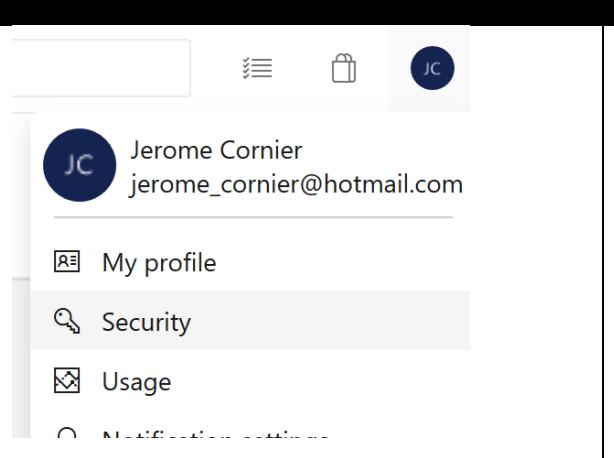
	<p>Parcourir la page Web jusqu'en bas</p> <p>Cliquer sur le fichier</p> <p>geckodriver-v0.24.0-win64.zip</p> <p>Save file</p> <p>Fermer Firefox</p>
	<p>Ouvrir l'Explorateur Windows</p> <p>Créer un répertoire bin dans C:\</p>
	<p>Ouvrir le fichier zip téléchargé dans Downloads</p> <p>Extraire (copier / coller) le fichier geckodriver dans le répertoire c:\bin</p>
	<p>Windows+Q</p> <p>Taper env</p> <p>Cliquer sur Edit the system environment variables</p>
	<p>Environnement Variables</p>

	<p>Sélectionner Path Edit</p>
	<p>New Ajouter C:\bin OK - OK</p>

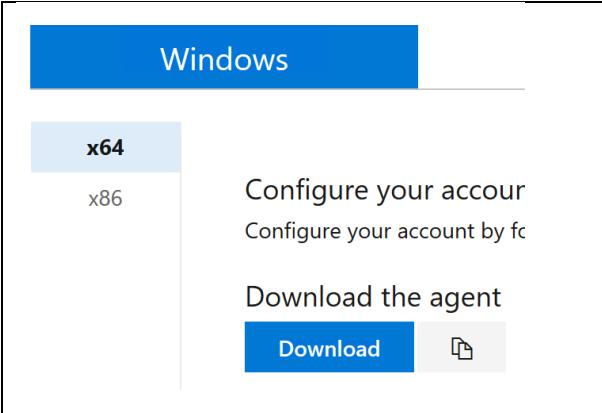
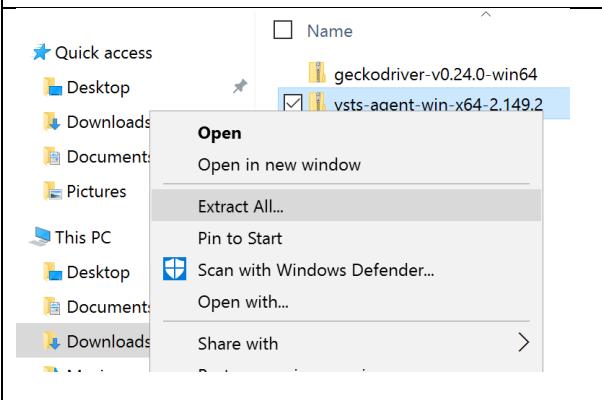
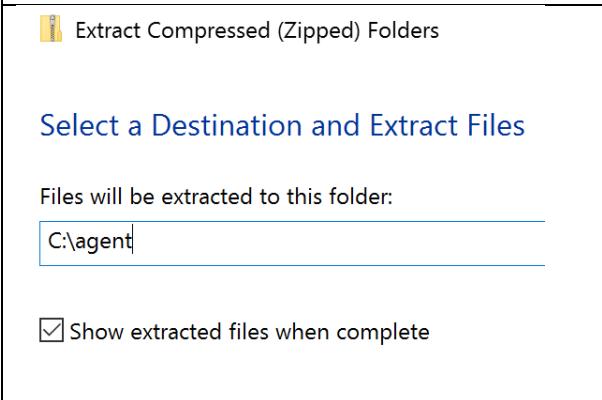
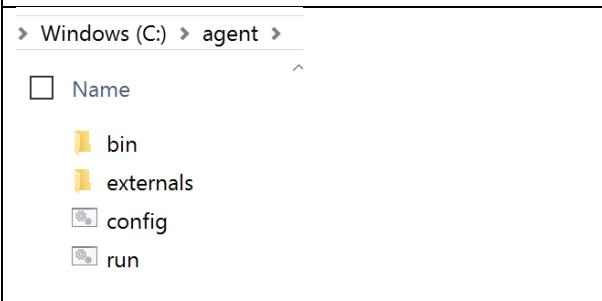
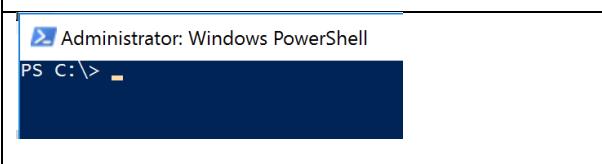
3.4 Installation de l'agent DevOps

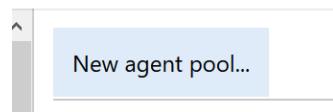
Référence :

<https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops>

Illustration	Action
	<p>Retourner sur la machine principale, dans DevOps, cliquer sur l'icône du profil Cliquer sur Security</p>
<p>Personal Access Tokens These can be used instead of a password + New Token</p>	<p>New Token</p>

<p>Name</p> <input type="text" value="A11yToken"/> <p>Organization</p> <input type="text" value="A11yDemo"/> <p>Expiration (UTC)</p> <input type="text" value="90 days"/>	<p>Choisir un nom</p> <p>Choisir une période de validité</p> <p>Full access</p> <p>Create</p> <p>Scopes</p> <p>Authorize the scope of access associated</p> <p>Scopes</p> <p><input checked="" type="radio"/> Full access <input type="radio"/> Custom defined</p>
<p>A11yToken token</p> <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;">fno477kofxxn7ucx</div>	<p>Copier et conserver le Token généré</p>
 Microsoft Azure  Azure DevOps	<p>Ouvrir la machine virtuelle de l'agent</p> <p>Lancer Firefox</p> <p>Taper l'adresse dev.azure.com</p> <p>Cliquer sur sign in</p> <p>Entrer les credentials</p>
<p>+ New organization</p> <p>⚙ Organization settings</p>	<p>Organization settings</p>
<p>Pipelines</p> <p>👤 Agent pools</p> <p>➡ Deployment pools</p>	<p>Agent pools</p>
<p>⬇ Download agent</p>	<p>Download agent</p>

	<p>Windows</p> <p>X64</p> <p>Download</p> <p>Save file</p>
	<p>Ouvrir l'explorateur de documents</p> <p>Sélectionner Downloads</p> <p>Cliquer droit sur le nouveau fichier téléchargé vsts-agent-win-x64xxx</p> <p>Extract All</p>
	<p>Entrer le répertoire de destination</p> <p>Extract</p>
	<p>Contrôler le répertoire c:\agent</p>
	<p>Lancer une fenêtre de commande PowerShell</p>

<pre>Administrator: Windows PowerShell PS C:\> cd .\agent PS C:\agent> .\config.cmd >> connect: Enter server URL > https://dev.azure.com/A11yDemo Enter authentication type (press enter for PAT) > Enter personal access token > ***** </pre>	<p>Aller dans le répertoire c:\agent Lancer le config.cmd Entrer l'URL de l'organisation DevOps créée précédemment Entrer pour l'option PAT (Token) Copier / coller le Token généré précédemment</p>
	<p>Aller dans DevOps New agent pool</p>
<p>Create an organization agent pool</p> <p>Name VMAgentA11y</p> <p><input checked="" type="checkbox"/> Auto-provision corresponding agent pools in all projects <input checked="" type="checkbox"/> Allow all pipelines to use this pool</p>	<p>Saisir le nom de l'agent pour créer un pool dédié OK</p>
<pre>Enter agent pool (press enter for default) > VMAgentA11y Enter agent name (press enter for VMAgentA11y) > Scanning for tool capabilities.</pre>	<p>Retourner dans la fenêtre de commande et saisir le nom du nouveau pool Entrer Entrer à nouveau pour le nom de l'agent par défaut</p>
<pre>Scanning for tool capabilities. Connecting to the server. Successfully added the agent Testing agent connection. Enter work folder (press enter for _work) > 2019-04-13 19:56:14Z: Settings saved. Enter run agent as service? (Y/N) (press enter for N) > Y</pre>	<p>Entrer pour le répertoire par défaut Y Entrer pour tourner l'agent en tant que service Entrer pour Network service par défaut</p>
<pre>D (press enter for N) > Y The service (press enter for NT AUTHORITY\SYSTEM) has started successfully. The service 'VMAgentA11y' successfully started. The service 'VMAgentA11y' successfully set recovery option. The service 'VMAgentA11y' successfully set to delayed auto-start. The service 'VMAgentA11y' successfully configured. The service 'VMAgentA11y' started successfully.</pre>	<p>L'agent doit démarrer</p>

The screenshot shows the 'Agents for pool VM Agent A11y' page in the Azure DevOps interface. On the left, there's a sidebar with 'All agent pools' listing various options like 'A11yVM', 'Default', 'Hosted', etc. In the center, a table lists agents under the 'Agents' tab. One agent, 'VM Agent A11y', is selected and highlighted with a green border. The table columns are 'Enabled', 'Name', 'State', and 'Current s'. The 'VM Agent A11y' row shows 'Enabled', 'VM Agent A11y', 'Online', and 'Idle'. At the bottom of the table, there's a button labeled '...'. The right side of the screen has two sections of text:

Aller dans DevOps

Contrôler la présence de l'agent dans le pool qui porte son nom

4 Implémentation de l'Agent DevOps A11y dans le pipeline

L'agent personnalisé étant opérationnel l'implémentation des tests d'accessibilité peut être réalisée directement dans le pipeline de livraison.

Sachant que l'agent fonctionne en analysant un site Web en ligne il faut déployer une première Web App pour procéder aux tests. C'est pourquoi nous avons préparé 2 Services d'application dans Azure.

Nous avons ajouté au pipeline 2 étapes :

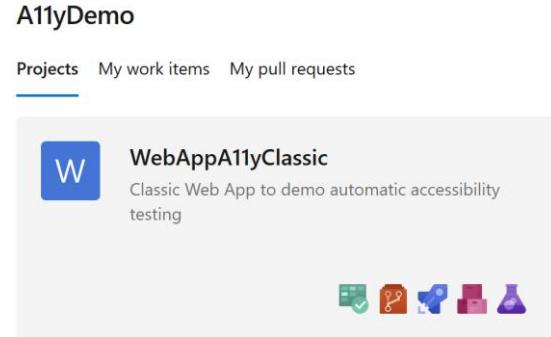
- Tests d'accessibilité sur la première Web App en ligne
- Si les tests sont passés avec succès, mise en ligne de la seconde Web App qui correspond au site Web de production

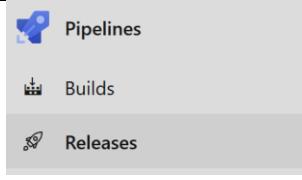
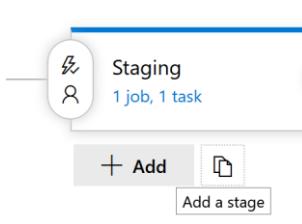
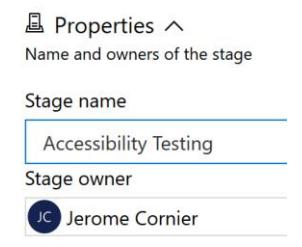
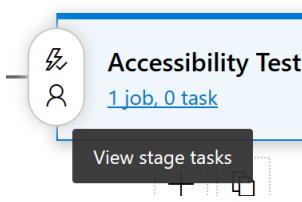
4.1 Implémentation de l'Agent dans le pipeline « Release »

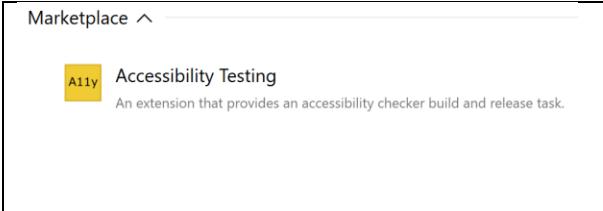
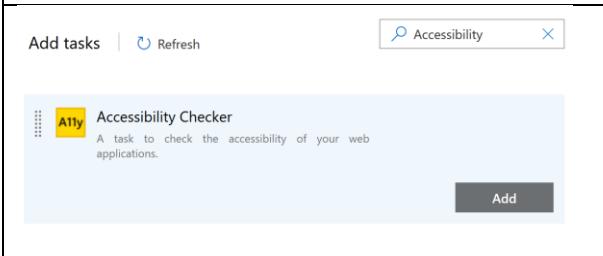
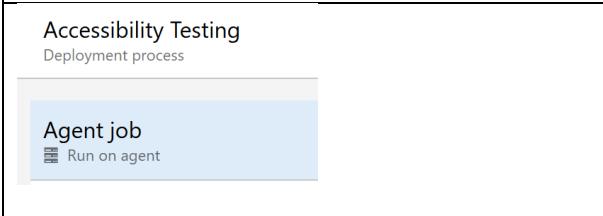
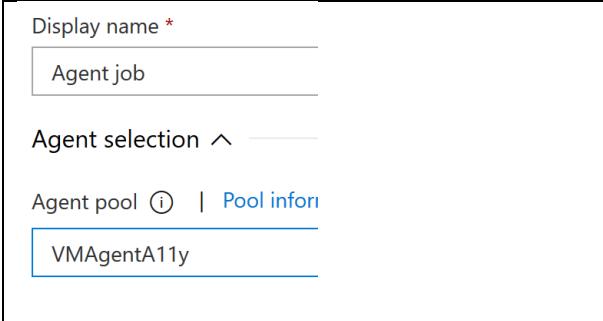
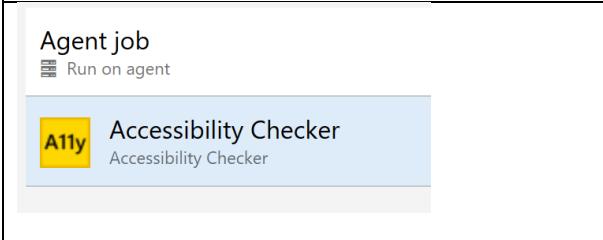
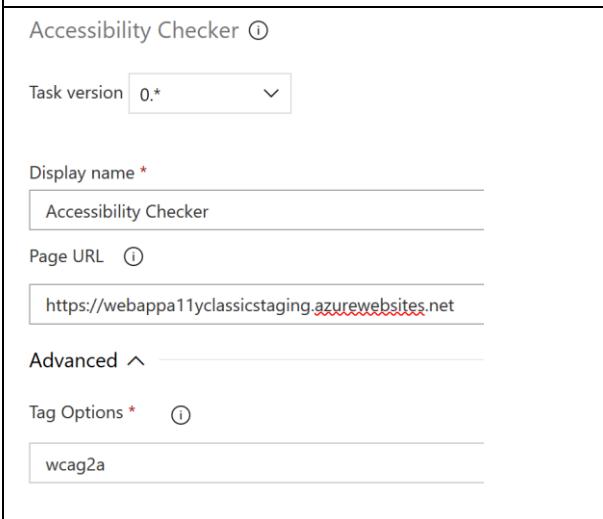
L'extension DevOps que nous utilisons prend en paramètre une URL. C'est cette page qui est analysée. Notre site Web possède plusieurs pages et nous voulons toutes les tester alors il faut ajouter autant de tâches de tests que de pages.

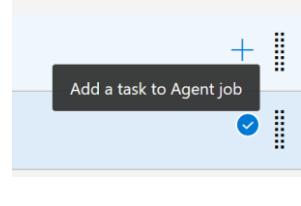
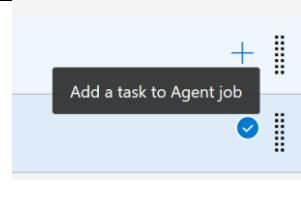
Le Référentiel d'accessibilité et le niveau sont paramétrables, nous avons choisi pour commencer le WCAG 2.0 niveau A.

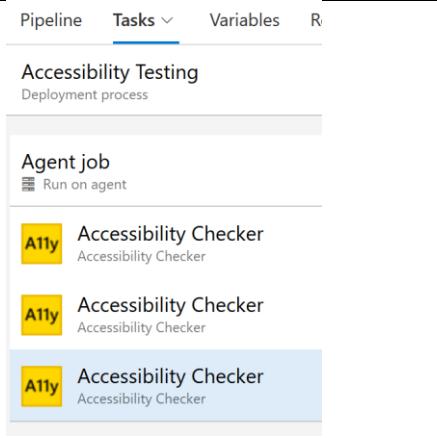
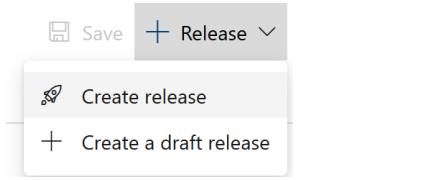
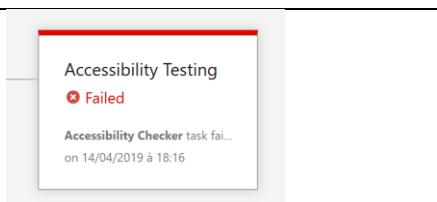
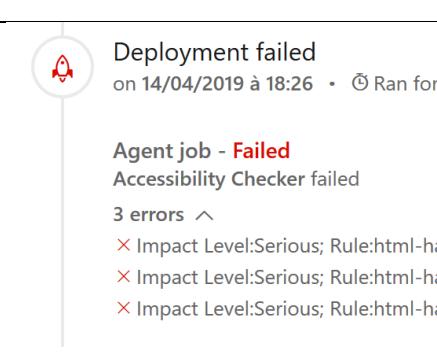
Il est possible de choisir d'interrompre le pipeline en cas d'erreur ou de continuer. Nous avons configuré les tests pour s'enchaîner sur toutes les pages et recevoir l'ensemble des non-conformités en une seule fois.

Illustration	Action
	<p>Aller dans DevOps</p> <p>Ouvrir le projet WebAppA11yClassic</p>

	Cliquer sur le pipeline Releases
	Cliquer sur Edit pour modifier le pipeline
	Positionner la souris sur Staging Cliquer sur Add (a stage) sous la boîte
Select a template Or start with an  Empty job	Cliquer sur Empty job
Stage Accessibility Testing 	Entrer le nom de l'étape
	Cliquer sur 1 job, 0 task
	Cliquer sur + sur la ligne Agent job
	Entrer Accessibility dans le champ de recherche

	<p>Accessibility Testing devrait apparaître</p> <p>Positionner la souris dessus</p> <p>Cliquer sur Install</p>
	<p>Accessibility Checker est disponible</p> <p>Positionner la souris dessus</p> <p>Cliquer sur Add</p>
	<p>Sélectionner Agent job</p>
	<p>Sélectionner l'Agent pool VMAgentA11y (c'est lui qui contient notre VM Custom)</p>
	<p>Sélectionner Accessibility Checker à gauche dans la liste des tâches de l'Agent job</p>
	<p>Mettre à jour l'URL à tester avec l'URL de la Web App Staging (copier / coller l'URL depuis Azure / Web App Staging)</p> <p>Déplier Advanced</p> <p>Contrôler wcag2a (garder ce paramètre pour l'instant)</p>

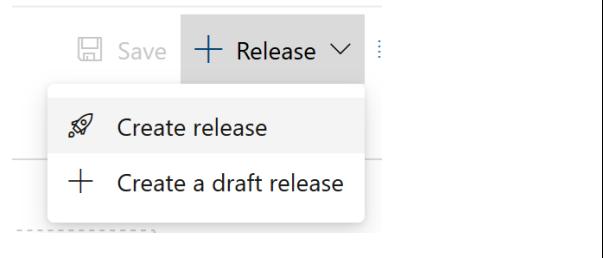
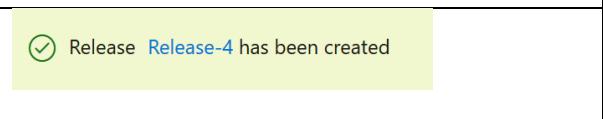
	<p>Ajouter une seconde tâche Accessibility Checker pour la seconde page</p>
<p>Display name *</p> <input type="text" value="Accessibility Checker"/> <p>Page URL ⓘ</p> <input type="text" value="https://webappa11classicstaging.azurewebsites.net/Privacy"/> <p>Advanced ▾</p> <p>Control Options ^</p> <p><input checked="" type="checkbox"/> Enabled</p> <p><input type="checkbox"/> Continue on error</p> <p>Timeout * ⓘ</p> <input type="text" value="0"/> <p>Run this task ⓘ</p> <p><input type="checkbox"/> Even if a previous task has failed, unless the deployment was canceled</p>	<p>Mettre à jour l'URL avec la page Privacy</p> <p>Déplier Control Options</p> <p>Changer le comportement: Even if a previous task has failed, unless the deployment was canceled</p>
	<p>Ajouter une troisième tâche Accessibility Checker pour la troisième page</p> <p>Page URL ⓘ</p> <input type="text" value="https://webappa11classicstaging.azurewebsites.net/Accessibility"/> <p>Advanced ▾</p> <p>Control Options ^</p> <p><input checked="" type="checkbox"/> Enabled</p> <p><input type="checkbox"/> Continue on error</p> <p>Timeout * ⓘ</p> <input type="text" value="0"/> <p>Run this task ⓘ</p> <p><input type="checkbox"/> Even if a previous task has failed, unless the deployment was canceled</p>

	La liste des tâches doit contenir 3 fois Accessibility Checker, 1 tâche par page Web du site
	Cliquer sur Save OK
	Cliquer sur Release Create release Create
	Cliquer sur le nom de la Release
	Cliquer sur Accessibility Testing (Failed, c'est normal)
	Le résultat devrait présenter des erreurs de type Impact Level :Serious ou Critical transmises par Accessibility Checker

4.2 Déploiement de l'application Web « Production »

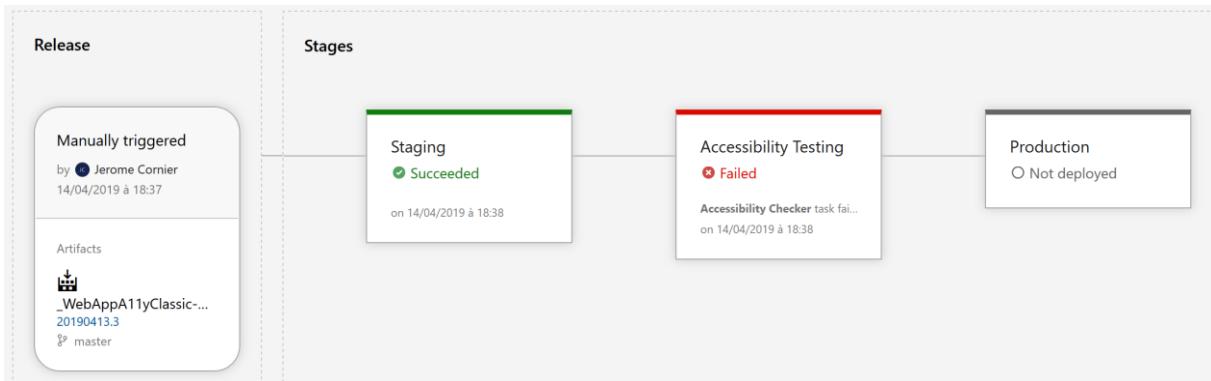
La dernière étape consiste à déployer en production si les tests sont réussis. Pour cela nous allons ajouter une étape au pipeline Release.

Illustration	Action
	Cliquer sur le nom du pipeline Release
	Cliquer sur Edit
	Positionner la souris sur Accessibility Testing et cliquer sur Add
	Positionner la souris sur Azure App Service deployment et cliquer sur Apply
	Renommer l'étape en Production
	Cliquer sur 1 job, 1 task
	Sélectionner la souscription Azure Conserver Web App on Windows Sélectionner WebAppA11yClassic (pas Staging, l'autre)
	Cliquer sur Save OK

	<p>Cliquer sur Release</p> <p>Create release</p> <p>Create</p>
	<p>Cliquer sur le nom de la Release</p>

4.3 Pipeline final

Voici à quoi ressemble le pipeline complet :



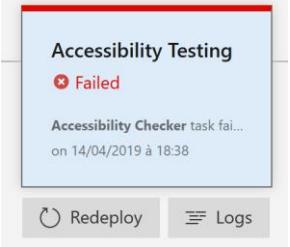
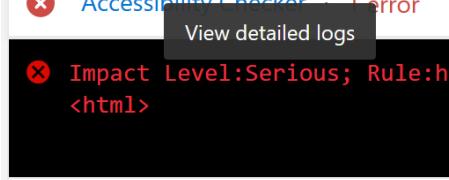
Pour cette Release certains tests ont échoué, la production n'a pas été mise en ligne. C'est précisément le comportement attendu en attendant les corrections pour la mise en conformité.

5 Démonstration des tests d'accessibilité automatisés

La plateforme de démonstration étant opérationnelles nous pouvons procéder à des modifications du code source dans Visual Studio pour valider le pipeline de livraison continue dans son intégralité.

5.1 Première Release en échec avec la Web App initiale

Nous l'avons évoqué précédemment, la solution initiale directement issue de Visual Studio comporte des erreurs de conformité.

Illustration	Action
	Cliquer sur Failed
	Cliquer sur Accessibility Checker
<pre>2019-04-14T16:38:37.0390880Z Rule:html-has-lang 2019-04-14T16:38:37.0390910Z Impact:Serious 2019-04-14T16:38:37.0390943Z Description:Ensures every HTML document has a lang attribute 2019-04-14T16:38:37.0390978Z More Info:https://dequeuniversity.com/rules/axe/3.1/html- has-lang?application=webdriverjs 2019-04-14T16:38:37.0457884Z ##[error]Impact Level:Serious; Rule:html-has-lang; <html> element must have a lang attribute <html></pre>	Dérouler jusqu'en bas le journal des erreurs Dans ce cas il s'agit du paramètre de lang qui est absent du _Layout.cshtml

Le pipeline fourni le détail de l'erreur dans le journal de l'agent DevOps. La solution de tests s'appuyant sur l'API axe-core, le journal donne directement accès aux détails en ligne concernant les erreurs.

5.1.1 Erreur n°1, paramètre de langue du document manquant

Pour la première erreur il s'agit du paramètre de langue du document qui est manquant.

How to Fix the Problem

Add a `lang` attribute to the `html` element (e.g. `<html lang="en">`) whose value represents the primary language of document.

Make sure you identify a language in the opening `<html>` element and spell the attribute correctly. For example, if the primary language of a document is English, you could specify the language as follows:

```
<html lang="en">
  <!--document head and body-->
</html>
```

Figure 4 : Correction proposée par Deque University

L'erreur est considérée « Sérieuse ». Le site Web de Deque University explique comment y remédier. En fin de page le lien vers la technique correspondante du W3C est disponible. Dans notre cas : [H57: Using language attributes on the html element](#).

5.1.2 Erreur n°2, texte alternatif manquent sur une image

Cette erreur est volontaire pour nos tests. Nous avons ajouté une page « Accessibility » au projet dans laquelle nous avons inséré le logo de l'accessibilité fourni par les Nations Unies.

Voici le code initial de la page ASP.NET dans lequel il manque en effet le texte alternatif.

```
@page
@model AccessibilityModel
 @{
     ViewData["Title"] = "Declaration of Accessibility";
 }



<h1>@ViewData["Title"]</h1>

    <p>This Demo Web Site is targeted to reach the Standard WCAG 2.0 A</p>

    <a href="https://www.un.org/fr/webaccessibility/logo.shtml">
        
    </a>
</div>


```

5.2 Mise en conformité du code d'après le rapport de tests

Nous pouvons corriger ces 2 erreurs, puis pousser les mises à jour dans GitHub ce qui a pour effet le déclenchement automatique du pipeline.

5.2.1 Correction du paramètre de langue

Le paramètre à ajouter se situe dans la page partagée nommée _Layout.cshtml.

```
<!DOCTYPE html>
<html lang="fr">
<body>
    [... contenu de la page ...]
</body>
</html>
```

5.2.2 Correction du texte alternatif de l'image

Même principe, nous pouvons ajouter le texte alternatif manquant.

```
<a href="https://www.un.org/fr/webaccessibility/logo.shtml">
    <img src "~/images/UN_Logo_Accessibility.jpg" alt="Accessibility Logo
created by United Nations" width="300" />
</a>
```

5.2.3 Synchronisation des modifications avec GitHub

L'opération peut se faire directement depuis Visual Studio en synchronisant le projet local avec GitHub pour récupérer les modifications ayant pu être effectuées en parallèle par un autre développeur, puis en poussant nos modifications.

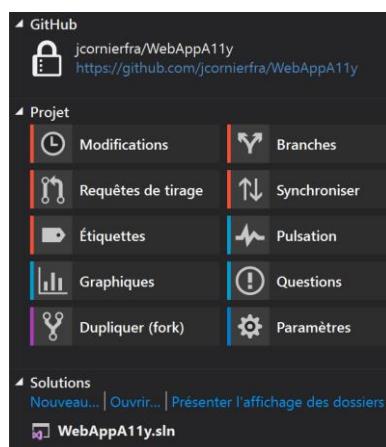


Figure 5 : Capture d'écran, Visual Studio Team Explorer synchronisé avec GitHub

5.3 Release réussie après correction

Après quelques minutes le pipeline nous donne le résultat de la dernière Release qui est maintenant conforme et déployé sur le Service d'application de production.

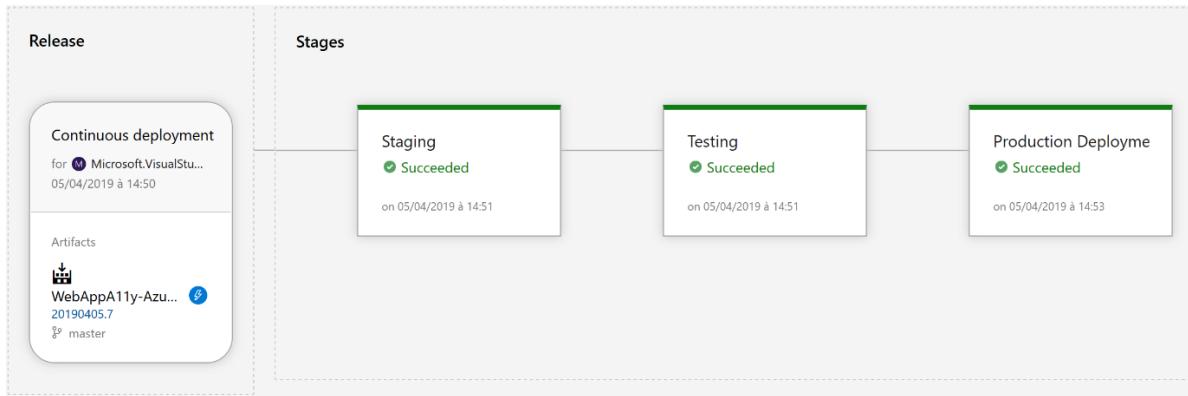


Figure 6 : Pipeline affichant le succès de toutes les étapes

Nous vérifions sur le site Web publié, le texte associé est reconnu par le navigateur.

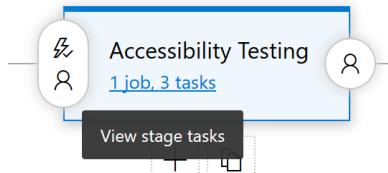
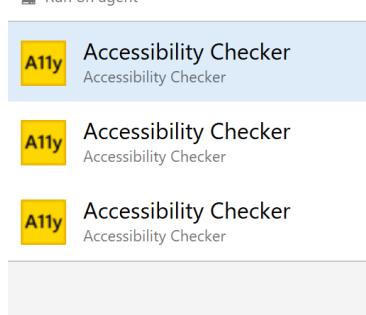
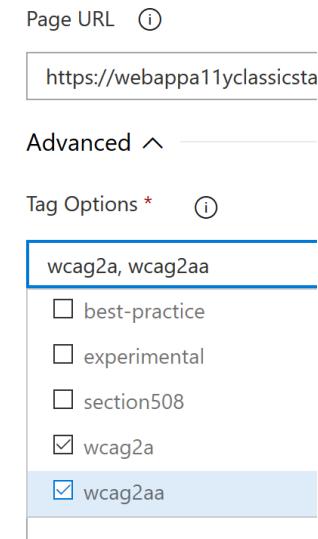
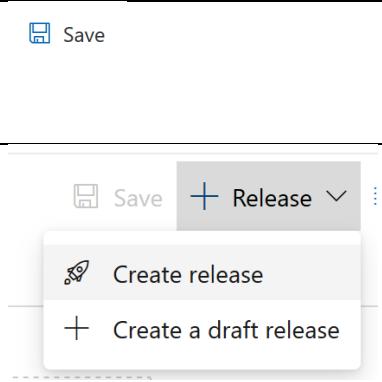


Figure 7 : Capture d'écran du site Web avec texte associé à l'image

5.4 Elévation du niveau d'accessibilité WCAG 2.0 A + AA

Les tests étant concluant nous allons augmenter le niveau des tests sur le site Web. Conformément à nos attentes l'élévation du niveau WCAG 2.0 A vers A et AA produit de nouvelles erreurs. Ce sont des erreurs de contraste, en effet le contraste est au niveau AA dans les référentiels d'accessibilité.

Illustration	Action
	Cliquer sur le nom du pipeline Release

	Cliquer sur Edit
	Cliquer sur 1 job, 3 Tasks dans l'étape Accessibility Testing
	Pour chaque Accessibility Checker, ajouter le wcag2aa
	Déplier Advanced Cocher wcag2aa (garder wcag2a)
	Cliquer sur Save OK Cliquer sur Release Create release Create

 Release Release-4 has been created	Cliquer sur le nom de la Release
--	---

Le pipeline devrait maintenant présenter des erreurs de contraste.

5.4.1 Identification des défauts de conformité

Pour faciliter l'identification de l'erreur dans la page nous pouvons utiliser le plugin WCAG 2.0 Contrast checker pour Firefox sur le poste de travail ou tout autre outil similaire.

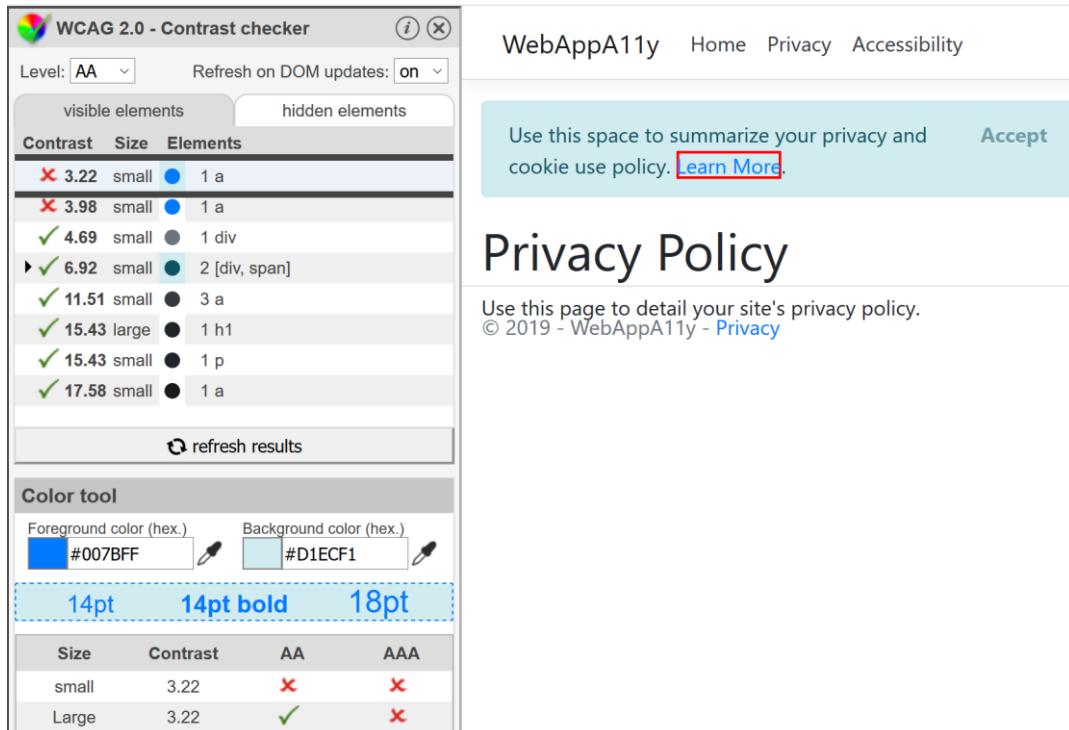


Figure 8 : Extension Color Contrast pour Firefox

Les erreurs concernent les 2 liens « Learn More » et « Privacy » qui s'affichent en effet en bleu moyen sur bleu clair ou blanc avec un contraste de 3,22 et 3,98, ce qui n'est pas très contrasté et inférieur au critère du WCAG 2.0 qui exige un contraste minimum de 4,5 pour les textes de petite taille.

5.4.2 Remédiation pour le niveau A + AA

Le site de Deque University vers lequel nous sommes orientés dans le journal d'erreur fourni aussi un outil d'analyse de contraste en ligne. Nous avons choisi de conserver la couleur de fond #D1ECF1 et le bleu du texte. Pour obtenir le contraste requis il faut foncer la couleur du texte, ce qui donne la valeur #0064CC.

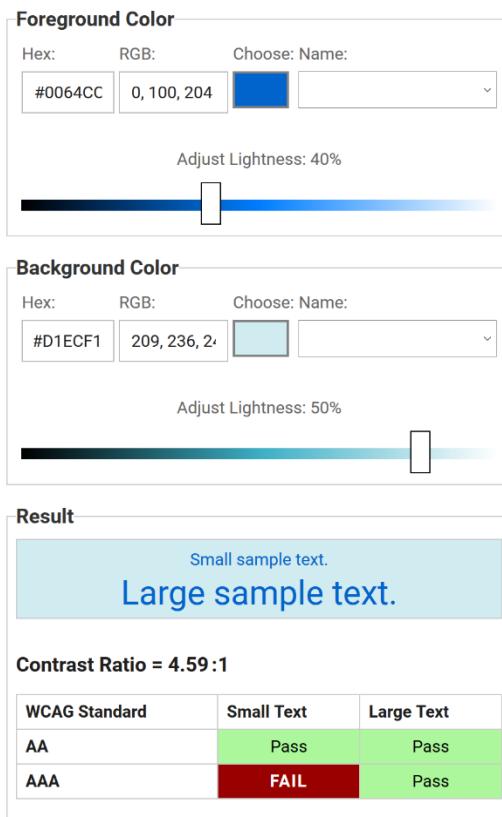


Figure 9 : Color Contrast Analyzer pour calculer la couleur nécessaire au contraste exigé

Nous apportons les modifications au code source, dans ce cas il faut ajuster les couleurs définies par la feuille CSS de la librairie Bootstrap dans le projet Visual Studio.

Notons que le choix de la nouvelle couleur répond au niveau AA mais pas au niveau AAA plus exigeant. C'est le genre de choix qu'il est important de définir dès la conception ou dans la charte de Design pour éviter les corrections au moment du développement.

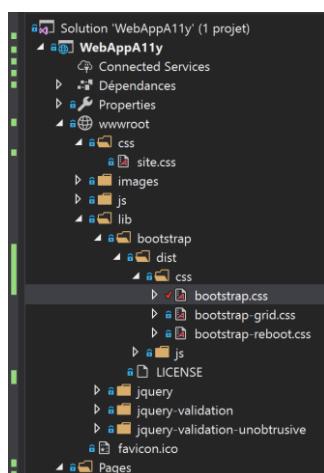


Figure 10 : Positionnement de la feuille CSS dans le projet Visual Studio

Bootstrap est une librairie additionnelle Open Source sujette à des mises à jour. Il est déconseillé de la modifier. La bonne pratique dans ce cas est de procéder à un CSS override,

c'est-à-dire l'ajout d'une nouvelle définition dans un autre fichier qui viendra se substituer à la définition originale. Il existe un fichier prévu pour cela dans le projet Visual Studio : site.css. Voici la modification apportée au code source.

```
...
/* Bootstrap.css override to darken the blue color of links
-----
a {
    /* color: #007bff; <- couleur bleue originale pour les liens */
    color: #0064cc; /* <- couleur bleue plus foncée conforme au WCAG 2.0 AA */
    text-decoration: none;
    background-color: transparent;
}
...
```

Les modifications sont poussées une nouvelle fois dans GitHub.



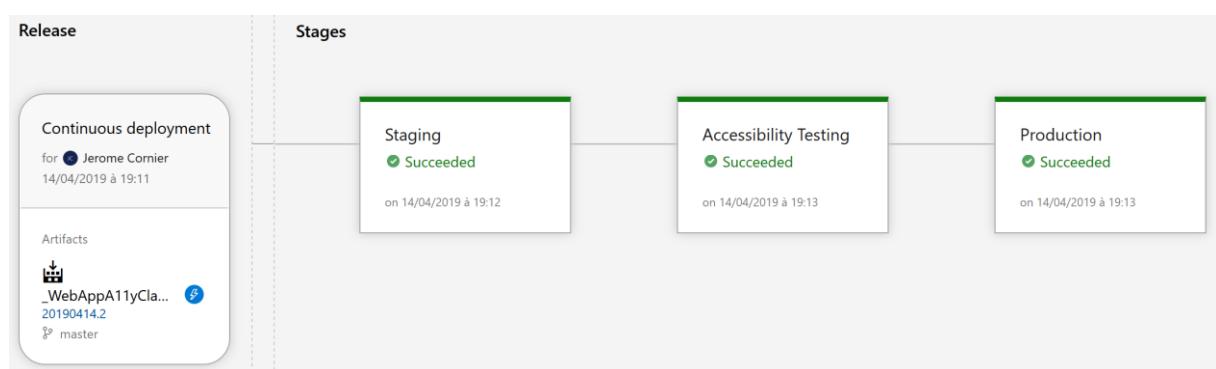
Figure 11 : Push des modifications de VS vers Github

Le pipeline « Builds » est automatiquement lancé.



Figure 12 : Pipeline « Builds » lancé automatiquement

Pipeline réussi.



5.4.3 Résultat avant / après

Les tests aux niveaux WCAG 2.0 A et AA sont passés et le résultat n'est finalement pas très éloigné de l'original à ceci près qu'il est désormais conforme aux critères d'accessibilité.

privacy and cookie use policy. [Learn More.](#)

privacy and cookie use policy. [Learn More.](#)

Welcome

Learn about [building Web apps with ASP.NET Core.](#)

Avant correction

Welcome

Learn about [building Web apps with ASP.NET Core.](#)

Avant correction

Figure 13 : Captures d'écran avant et après pour illustrer la variation de couleur pour passer le niveau AA

6 Web App en Container – Préparation de l'environnement

Le second prototype que nous avons construit met en œuvre des containers Docker pour héberger les deux sites Web de préproduction et de production.

6.1 Architecture générale

Pour cette version de l'application en container Docker nous utiliserons la convention de noms suivante. Les principales différences par rapport à la version classique sont :

- Azure Container Repository pour stocker l'image du container
- Les App Services qui deviennent des Web App for Containers

Nom	Composant
Application Web	
WebAppA11yDocker	Solution Visual Studio
WebAppA11yDocker	Référentiel GitHub
Ressources Azure	
WebAppA11yDockerRG	Resource group
WebAppA11yDockerACR	Azure Container Repository
WebAppA11yDockerStaging	Web App for Containers « Staging »
WebAppA11yDockerStagingPlan	App Service Plan « Staging »
WebAppA11yDocker	Web App for Containers « Production »
WebAppA11yDockerPlan	App Service Plan « Production »
Azure DevOps	
A11yDemo	Organisation DevOps
WebAppA11yDocker-ASP.NET-CI	Pipeline « Builds »
WebAppA11yDocker-ASP.NET-CD	Pipeline « Release »

6.1.1 Architecture de la Web App Linux en Container

L'architecture diffère de l'application Web précédente car il faut prendre en compte :

- Le container Docker qui doit être stocké dans un Repository. Nous avons choisi Azure Container Registry qui suffit pour ce prototype simple
- Les services d'application qui doivent s'appuyer sur les containers, sachant que ceux-ci ont un système d'exploitation Linux

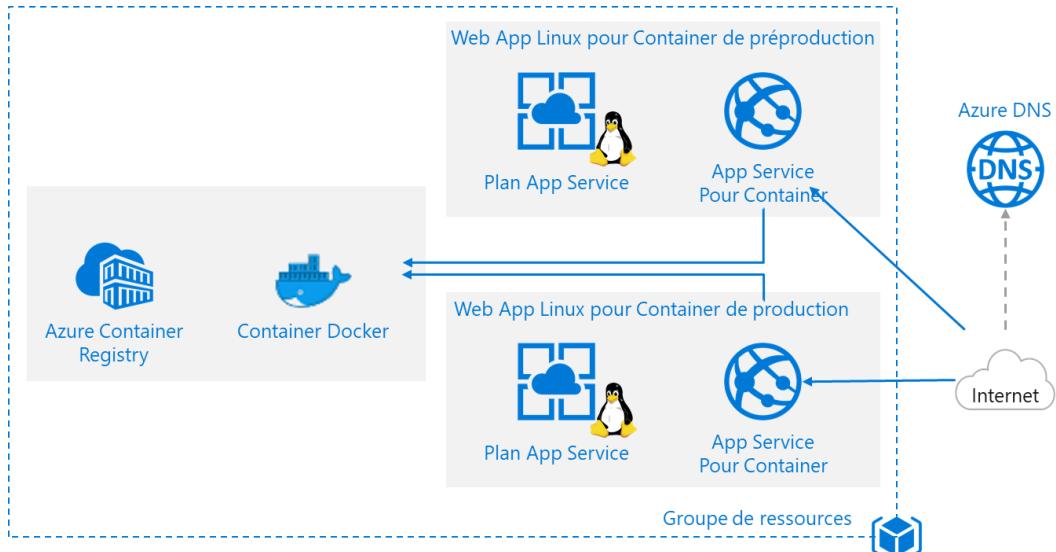


Figure 14 : Architecture Web App pour Container Docker

Nous aurions pu utiliser Azure Kubernetes Services au lieu de Azure Container Registry mais cela n'était pas justifié et surtout l'architecture AKS nécessite minimum 2 machines virtuelles puissantes qui portent le coût mensuel au-delà de notre souscription.

6.1.2 Architecture du pipeline pour le container Docker

La présentation du pipeline pour la solution en container paraît plus complexe car elle fait appel à des éléments d'architecture complémentaires mais en pratique elle se révèle plus rapide, avec moins d'étapes pour le pipeline « Builds ».

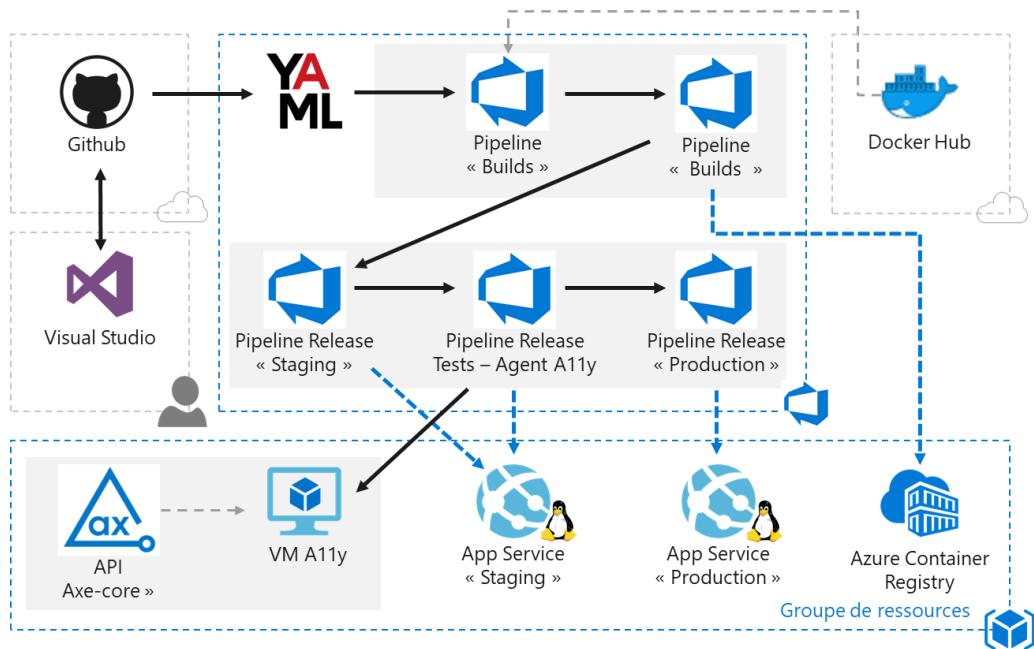


Figure 15 : Pipeline DevOps pour une application Web en Container

Les principales différences avec la solution précédente sont :

- Un fichier YAML de script du pipeline « Builds » est nécessaire
- Le pipeline « Builds » doit tirer une image de base depuis Docker Hub pour la construire avec l'application Web
- Le déploiement n'est plus le chargement du site Web dans le Service d'application mais l'envoi du nouveau container dans Azure Container Registry

Les tests d'accessibilité automatisés quant à eux sont identiques puisqu'ils utilisent l'URL du site Web publié.

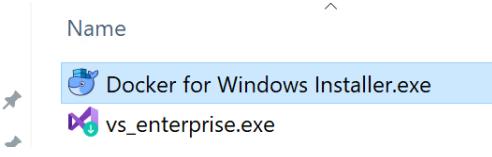
6.2 Ajout de Docker à l'environnement de développement

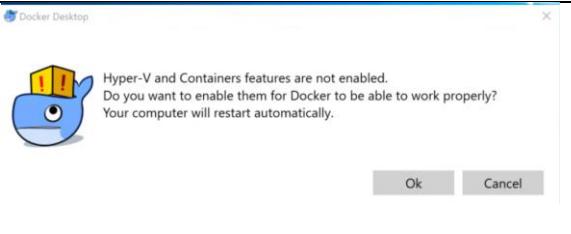
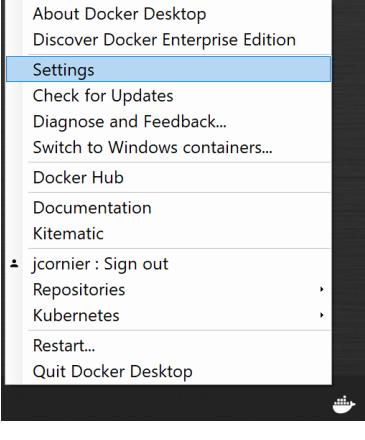
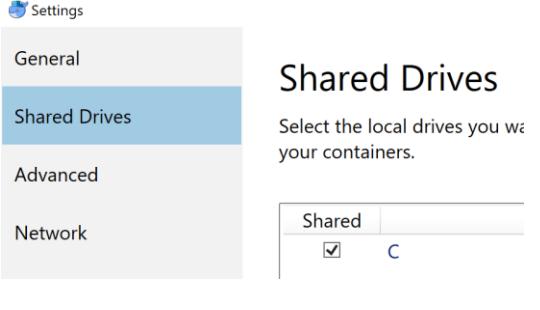
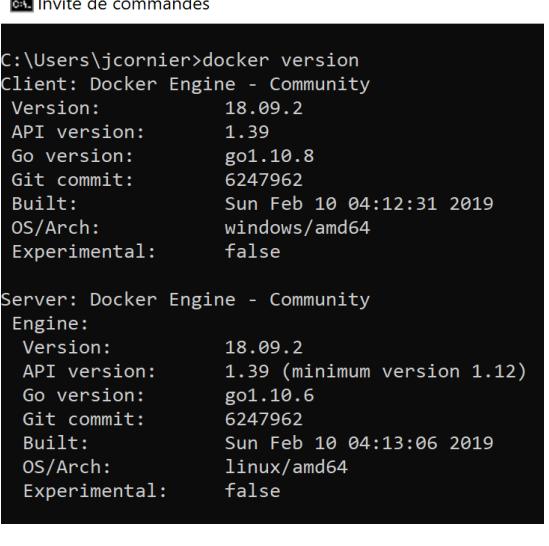
Docker Desktop doit être installé sur le poste de travail. L'installation de cette application fournit le nécessaire pour créer des projets Visual Studio avec le support Docker et pour tester en local la solution.

Une extension des commandes en ligne permet de gérer les containers sur le poste local.

6.2.1 Installation de Docker sur le poste de développement

Illustration	Action
--------------	--------

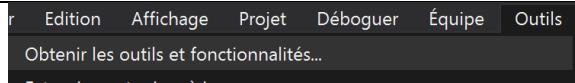
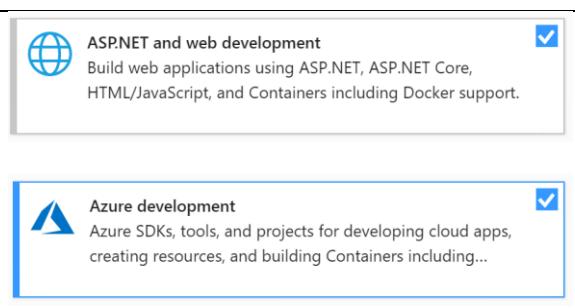
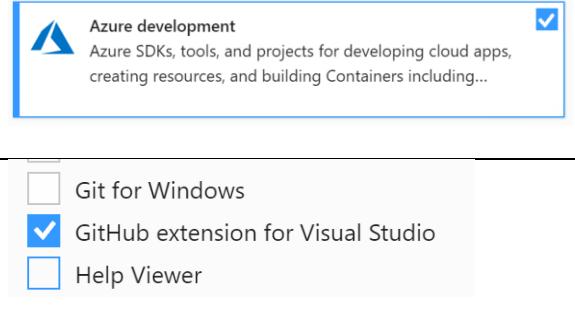
<h2>Docker Desktop</h2> <p>The preferred choice for millions of developers that are building containerized applications</p> <p>Download for Mac Download for Windows</p>	<p>Ouvrir un navigateur Taper l'adresse : https://www.docker.com/products/docker-desktop cliquer sur Download for Windows</p>
<p>Get Docker Desktop for Windows</p> <p>Docker Desktop for Windows is available for free.</p> <p>Requires Microsoft Windows 10 Professional or Enterprise 64-bit. For previous versions get Docker Toolbox.</p> <p>By downloading this, you agree to the terms of the Docker Software End User License Agreement and the Docker Data Processing Agreement (DPA).</p> <p>Get Docker</p>	<p>Se connecter ou créer un compte si besoin Cliquer sur Get Docker Save</p>
<p>> This PC > Downloads</p> 	<p>Aller dans le répertoire de téléchargement Lancer l'exécutable d'installation de Docker</p>
<p> Installing Docker Desktop</p> <h3>Configuration</h3> <p><input checked="" type="checkbox"/> Add shortcut to desktop <input type="checkbox"/> Use Windows containers instead of Linux containers</p>	<p>Décocher Use Windows containers</p>
<p> Installing Docker Desktop</p> <p>Docker Desktop 2.0.0.3</p> <p>Installation succeeded</p> <p>You must log out of Windows to complete installation.</p> <p>Close and log out</p>	<p>L'installation nécessite une déconnexion / reconnexion</p>

	<p>Après la reconnexion Docker demande l'activation des fonctionnalités.</p> <p>Cliquer sur OK</p>
	<p>Cliquer droit sur l'icône Docker dans la barre de notification</p> <p>Cliquer sur Settings</p>
	<p>Sélectionner Shared Drives</p> <p>Vérifier qu'un disque est partagé</p> <p>Fermer la fenêtre des paramètres</p>
 <pre>C:\Users\jcornier>docker version Client: Docker Engine - Community Version: 18.09.2 API version: 1.39 Go version: go1.10.8 Git commit: 6247962 Built: Sun Feb 10 04:12:31 2019 OS/Arch: windows/amd64 Experimental: false Server: Docker Engine - Community Engine: Version: 18.09.2 API version: 1.39 (minimum version 1.12) Go version: go1.10.6 Git commit: 6247962 Built: Sun Feb 10 04:13:06 2019 OS/Arch: linux/amd64 Experimental: false</pre>	<p>Démarrer une invite de commandes</p> <p>Taper docker version</p> <p>La liste des composants et des versions doit apparaître</p> <p>Fermer l'invite de commandes</p>

6.2.2 Ajout du support ASP.NET et Docker dans Visual Studio

Illustration	Action
--------------	--------

Comment implémenter les pratiques inclusives
dans le Continuous Delivery Pipeline ?

	<p>Démarrer Visual Studio</p> <p>Cliquer sur Outils, puis Obtenir les outils de fonctionnalités</p>
	<p>Dans « Charges de travail » vérifier ou cocher que ASP.NET et Azure development sont cochés</p>
 <ul style="list-style-type: none"> <input type="checkbox"/> Git for Windows <input checked="" type="checkbox"/> GitHub extension for Visual Studio <input type="checkbox"/> Help Viewer 	<p>Dans Composants individuels, vérifier ou cocher GitHub extension for Visual Studio</p>

6.3 Crédit de l'application Web en container et publication dans GitHub

Pour cette préparation nous procéderons en 6 étapes :

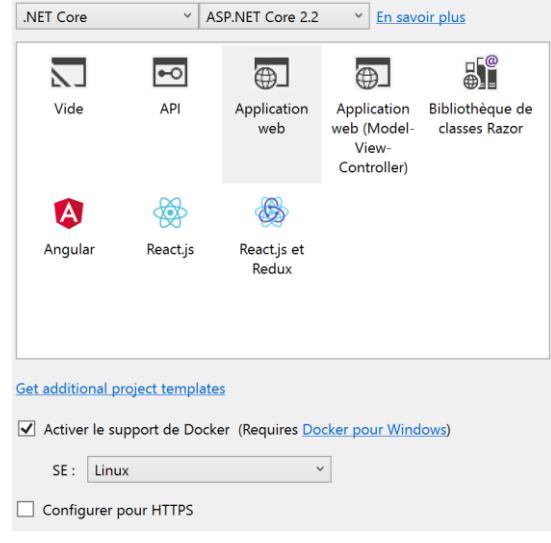
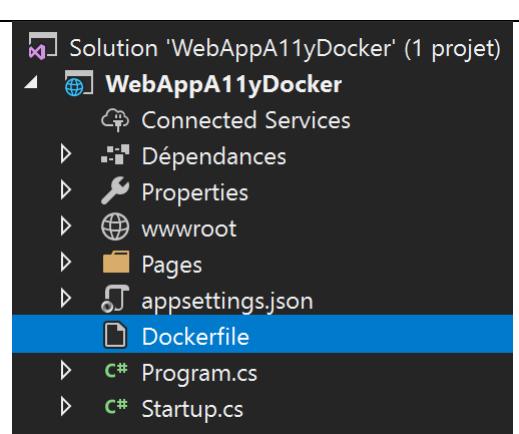
- Crédit de l'application pour Docker dans Visual Studio
- Modifier le fichier Dockerfile
- Test avec Docker en local
- Publication dans GitHub
- Crédit d'une nouvelle page dans le projet de l'application
- Contrôle de mise à jour dans GitHub

6.3.1 Crédit de l'application dans Visual Studio

Cette opération est similaire à celle déjà effectuée pour l'application Web classique. La principale différence est la prise en charge du container Docker dans le projet Visual Studio.

Illustration	Action

	Lancer Visual Studio Créer un nouveau projet
	Sélectionner .NET Core
	Sélectionner Application web ASP.NET Core
<p>Nom : <input type="text" value="WebAppA11yDocker"/></p> <p>Emplacement : <input type="text" value="C:\Users\jcornier\source\repos\jcornierfra\"/></p> <p>Nom de solution : <input type="text" value="WebAppA11yDocker"/></p>	<p>Saisir le nom de l'application</p> <p>Ne nom de la solution (le même)</p> <p>Choisir le répertoire</p>
<input type="button" value="Parcourir..."/> <input checked="" type="checkbox"/> Créer un répertoire pour la solution <input type="checkbox"/> Créer un dépôt Git	<p>Cocher Créer un répertoire pour la solution</p> <p>Ne pas cocher Créer un dépôt Git (nous utiliserons GitHub)</p> <p>Valider ces choix par OK</p>

	<p>Vérifier que Application web est sélectionné</p> <p>Décocher HTTPS</p> <p>Cocher « Activer le support Docker »</p> <p>SE : Linux</p> <p>OK</p>
	<p>Le projet apparaît dans Visual Studio / Explorateur de solutions</p> <p>Le fichier Dockerfile apparaît, c'est une différence par rapport à l'application classique</p>

6.3.2 Modifier le fichier Dockerfile

Le fichier Dockerfile créé par défaut ne fonctionne pas, il faut l'adapter à la structure de l'application. Voici le fichier que nous proposons pour prototype.

```
FROM microsoft/dotnet:2.2-aspnetcore-runtime AS base
WORKDIR /app
EXPOSE 80

FROM microsoft/dotnet:2.2-sdk AS build
WORKDIR /app

# Build solution
COPY . .
RUN dotnet restore

RUN dotnet build -c Release -o out

FROM build AS publish
RUN dotnet publish -c Release -o out

# Build runtime image
```

```

FROM base AS final
WORKDIR /app
COPY --from=publish /app/out ./
ENTRYPOINT ["dotnet", "WebAppA11yDocker.dll"]

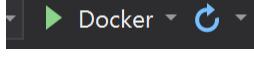
```

Ce script montre les opérations suivantes qui devront être réalisées pour construire le container avec l'application Web :

- Chargement de l'image aspnetcore-runtime en dotnet 2.2 qui contiendra l'application Web dans le répertoire /app et qui sera exposée sur le port http 80
- Chargement de l'image sdk en dotnet 2.2 pour construire et publier l'application Web
- Les sources sur projet sont copiées dans le répertoire /app de l'image sdk
- Dotnet restore permet de reconstruire le projet en chargeant le fichier de solution .sln
- L'application est construite puis publiée
- Les fichiers résultants sont ensuite copiés de l'image sdk vers l'image runtime
- L'application est devenue une librairie .dll qui sert de point d'entrée dans le container

Le principe est le même que ce soit sur le poste local ou dans le pipeline de livraison continue. Il peut exister quelques différences, par exemple lorsque plusieurs containers doivent exposer les sites sur des ports spécifiques, dans ce cas un fichier Dockerfile.override permet de modifier la création du container selon l'usage pour des tests locaux ou pour l'envoie dans le pipeline.

6.3.3 Test de l'application Docker en local

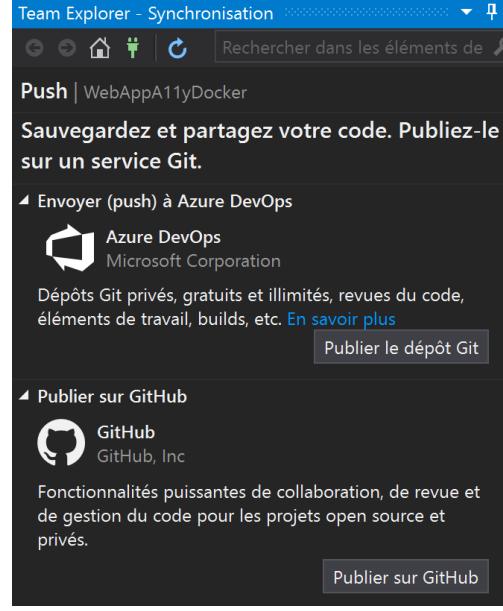
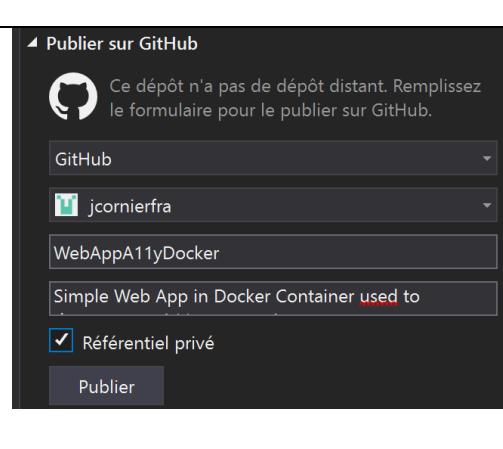
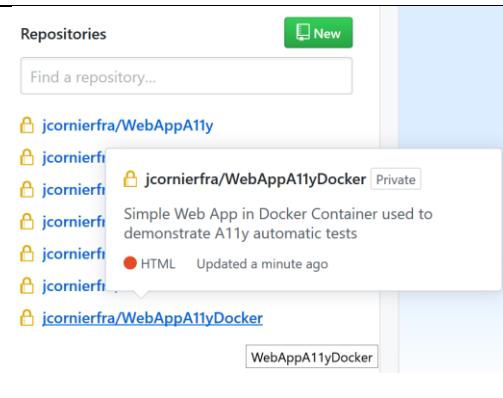
Illustration	Action
	Cliquer droit sur la solution, puis Générer la solution pour vérifier la construction sans erreur
===== Génération : 1 a réussi, 0 a échoué, 0 mis à jour, 0 a été ignoré =====	Le résultat doit être réussi
	Cliquer sur Docker

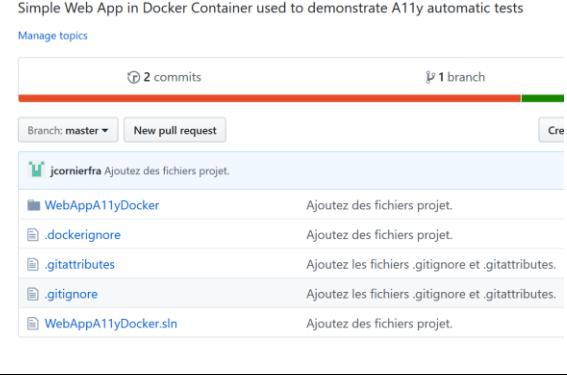
	Après quelques instants le site web est ouvert dans un navigateur. localhost :xxxxx correspond à la machine locale qui expose le site sur un port étendu (le port 80 est généralement déjà utilisé)
	Cliquer sur le carré rouge pour stopper le service
<pre>C:\>docker container ps CONTAINER ID IMAGE COMMAND NAMES 53a5855d8d01 webappa11ydocker:dev "tail -f /dev/n 80/tcp trusting_benz C:\>-</pre>	Ouvrir une invite de commandes (pas besoin du mode admin) Taper docker container ps Le container webappa11ydocker apparaît
<pre>C:\>docker container ps CONTAINER ID IMAGE COMMAND NAMES 53a5855d8d01 webappa11ydocker:dev "tail -f /dev/n 80/tcp trusting_benz C:\>docker container stop 53a5855d8d01</pre>	Taper docker container stop <container ID> pour arrêter le service Container ID peut être copier/coller depuis la liste précédente Taper Exit pour fermer l'invite de commandes

6.3.4 Publication de l'application dans GitHub

Il s'agit de la même opération que pour l'application classique.

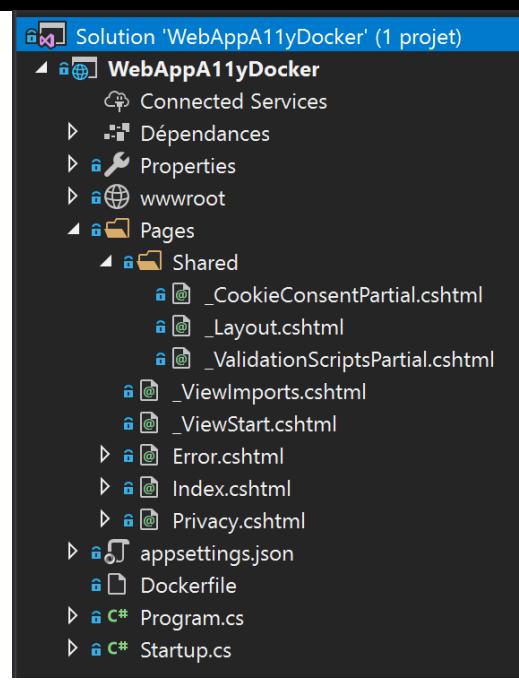
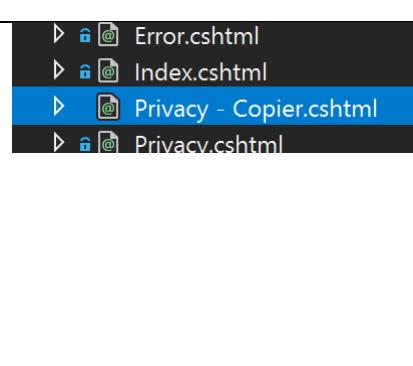
Illustration	Action
	Cliquer en bas à droite sur Ajouter au contrôle de code source Cliquer sur Git

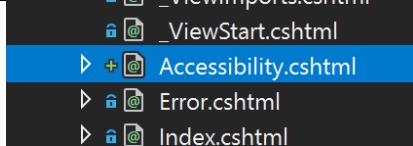
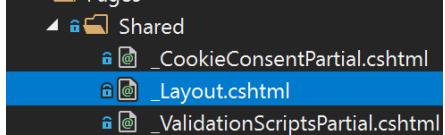
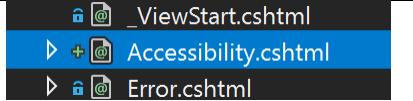
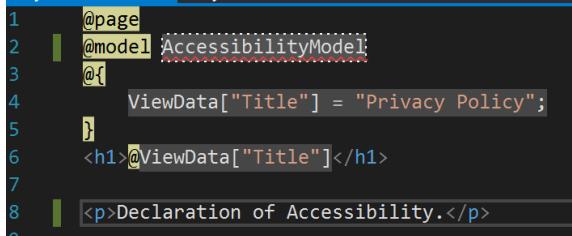
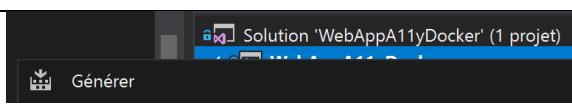
	<p>Cliquer sur Publier sur GitHub</p> <p>ATTENTION Azure DevOps est un autre dépôt. Nous voulons sélectionner GitHub</p>
	<p>Vérifier ou sélectionner le compte/profile GitHub</p> <p>Saisir le nom de la Web App</p> <p>Saisir une courte description (modifiable ultérieurement)</p> <p>Cocher Référentiel privé</p> <p>Publier</p>
	<p>Le dépôt a été créé</p>
	<p>Ouvrir https://github.com</p> <p>Cliquer sur le nom de la Web App</p>

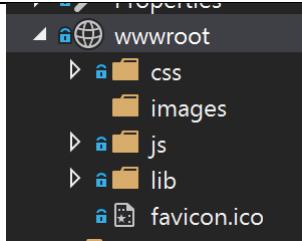
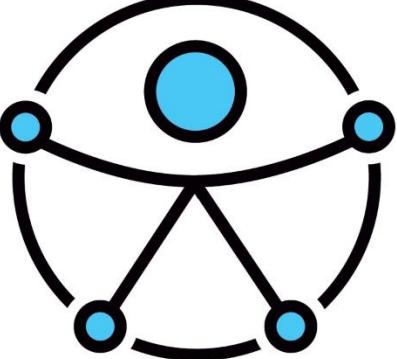
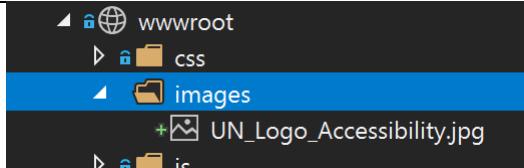
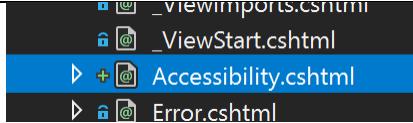
	<p>Les fichiers sont présents et le fichier <code>.dockerignore</code> a été ajouté.</p>
---	--

6.3.5 Ajout d'une page Web dans l'application

Les modifications que nous proposons sont les mêmes que pour l'application classique. Il n'y a aucune différence dans le code lui-même.

Illustration	Action
	<p>Revenir dans Visual Studio, Explorateur de solutions</p> <p>Déployer Pages et Shared</p>
	<p>Cliquer bouton droit sur Privacy.cshtml</p> <p>Copier</p> <p>Cliquer bouton droit sur Pages</p> <p>Coller</p> <p>Le nouveau fichier Privacy-Copier apparaît</p>

	Renommer en Accessibility.cshtml
	Dans Shared, cliquer sur _Layout.cshtml Le contenu s'ouvre dans la fenêtre centrale
	Sélectionner et copier les 3 lignes correspondant à Privacy (Ctrl-C, Ctrl-V, Ctrl-V)
<pre>asp-page="/Index">Home asp-page="/Privacy">Privacy asp-page="/Accessibility">Accessibility</pre>	Renommer Privacy en Accessibility sur la nouvelle ligne Enregistrer (Ctrl-S)
	Sélectionner Accessibility.cshtml
	Remplacer Privacy par Accessibility Enregistrer
	Déplier Accessibility et sélectionner Accessibility.cshtml.cs
	Renommer PrivacyModel en AccessibilityModel Enregistrer
	Cliquer bouton droit sur la Solution Générer la solution

===== Génération : 1 a réussi, 0 a échoué, 0 mis à jour, 0 a été ignoré =====	La solution devrait se construire sans erreur
	Cliquer bouton droit sur wwwroot Ajouter Nouveau dossier Saisir le nom « images » Valider
	Enregistrer l'image ci-contre dans un fichier jpeg sous le nom UN_Logo_Accessibility.jpg
	Glisser et déposer le fichier image dans le dossier images
	Cliquer sur Accessibility.cshtml

Ajouter le logo dans le code qui doit ressembler à celui-ci-dessous. Il manque le alt=« xxx » pour l'image, c'est volontaire pour les tests d'accessibilité.

```

@page
@model AccessibilityModel
 @{
   ViewData["Title"] = "Declaration of Accessibility";
 }

<div class="text-center">
  <h1>@ViewData["Title"]</h1>

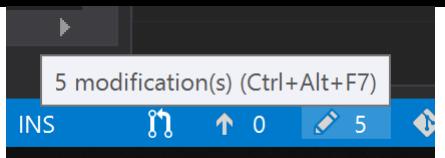
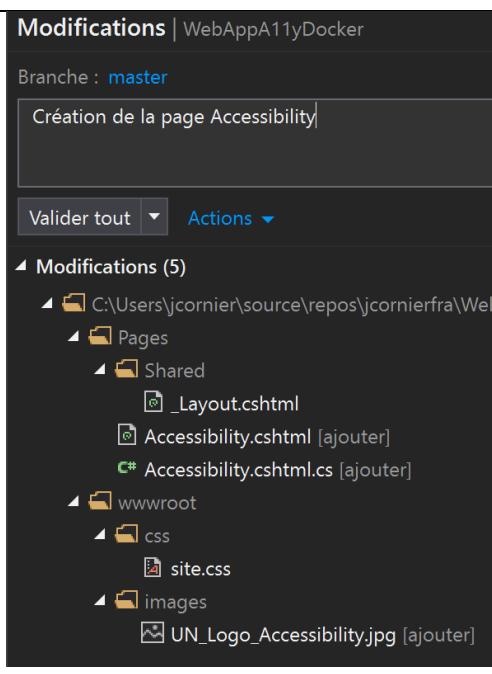
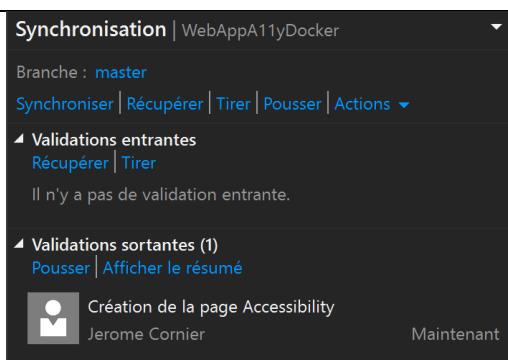
  <p>This Demo Web Site is targeted to reach the Standard WCAG 2.0 A + AA</p>

  <a href="https://www.un.org/fr/webaccessibility/logo.shtml">
    <img src "~/images/UN_Logo_Accessibility.jpg" width="300" />
  </a>

```

</div>

6.3.6 Mise à jour dans GitHub

Illustration	Action
	Cliquer sur le stylo en bas à droite (Autre solution : dans Team Explorer / Modifications)
	Entrer une description correspondant aux modifications effectuées Vérifier la liste des fichiers qui correspond à ceux modifiés précédemment Cliquer sur Valider tout
	Cliquer sur Synchroniser
	Cliquer sur Pousser

Synchronisation WebAppA11yDocker	La synchronisation a été effectuée
ⓘ Push effectué correctement vers origin/master.	

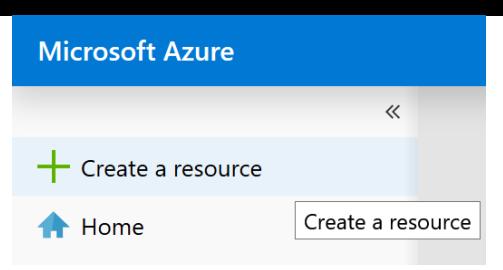
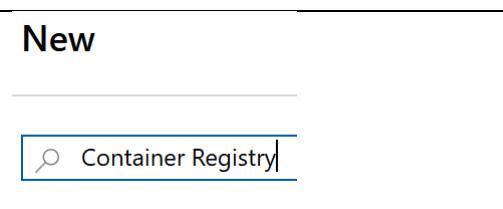
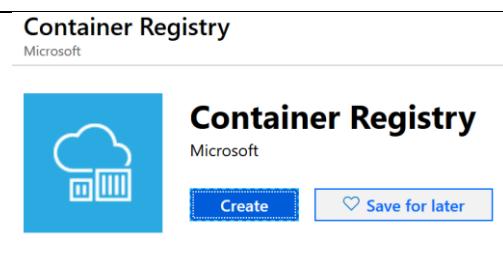
6.4 Création de l'environnement Azure pour la Web App en Container

L'étape suivante consiste à préparer l'environnement Cloud pour recevoir l'application.

A partir du portail Azure (<http://portal.azure.com>) nous devons créer :

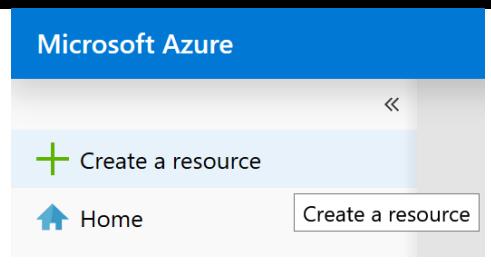
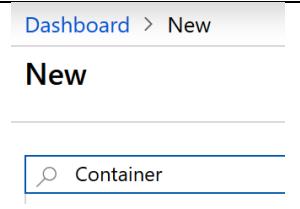
- 1 référentiel de container, Azure Container Repository
- 2 Services d'application Web App pour Container avec leurs plans de services associés : la première Web App servira de préproduction « Staging » pour tester l'accessibilité et la seconde sera déployée en production si les tests sont réussis

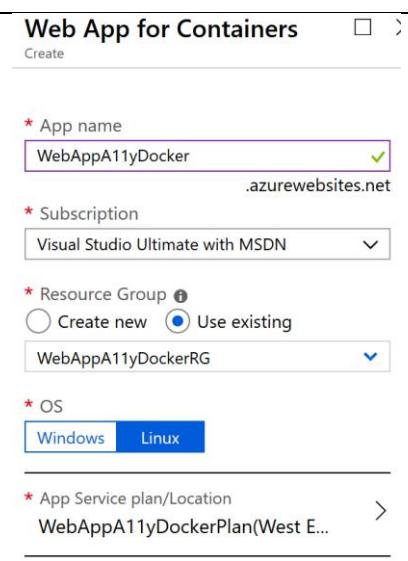
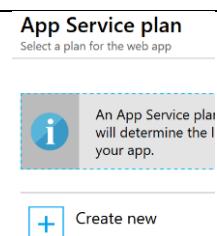
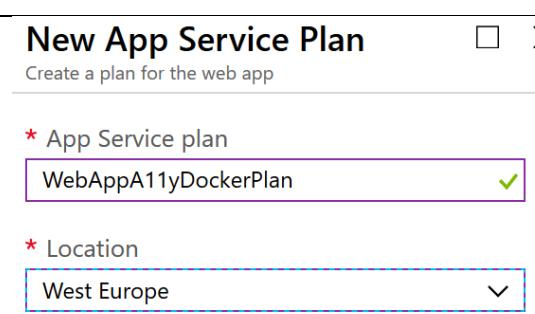
6.4.1 Création du référentiel de containers

Illustration	Action
	Dans le portail Azure, cliquer sur Create a resource
	Taper Container Registry dans la barre de recherche, puis retour
	Create

<p>Create container registry</p> <p>* Registry name WebAppA11yDockerACR.azurecr.io</p> <p>* Subscription Visual Studio Ultimate with MSDN</p> <p>* Resource group WebAppA11yDockerRG Create new</p> <p>* Location West Europe</p> <p>* Admin user i Enable Disable</p> <p>* SKU i Standard</p>	<p>Nommer le registre, nom de la Web App suivi de ACR</p> <p>Sélectionner la souscription Azure</p> <p>Créer un nouveau groupe de ressources, nom de la Web App suivi de RG</p> <p>Create</p>
<p> Deployment succeeded</p> <p>Deployment 'Microsoft.ContainerRegistry' to resource 'WebAppA11yDockerRG' was successful.</p> <p>Go to resource Pin to dashboard</p>	<p>Pin to dashboard</p>

6.4.2 Crédit des Web App for Containers

Illustration	Action
	<p>Dans le portail Azure, cliquer sur Create a resource</p>
	<p>Taper container dans la barre de recherche, puis retour</p>

 Web App for Containers	Cliquer sur Web App for Containers
	Create
	<p>Saisir le nom de la Web App</p> <p>Choisir la souscription (si plusieurs)</p> <p>Utiliser le groupe de ressource créé précédemment</p> <p>Sélectionner Linux</p> <p>Cliquer sur App Service plan</p>
	Create new
	<p>Saisir le nom suivi de « Plan »</p> <p>Choisir l'emplacement (« West Europe » dans notre exemple)</p> <p>Cliquer sur Pricing tier pour changer le Plan par défaut en B1</p>

 <p>Dev / Test</p> <p>For less demanding workloads</p> <p>for Linux is free for the first 30 days!</p> <p>Recommended pricing tiers</p> <table border="1" style="background-color: #6aa84f; color: white;"> <tr> <td>B1</td> <td>100 total ACU 1.75 GB memory A-Series compute equivalent 33.18 EUR/Month (Estimated)</td> </tr> </table>	B1	100 total ACU 1.75 GB memory A-Series compute equivalent 33.18 EUR/Month (Estimated)	<p>Cliquer sur Dev / Test</p> <p>Cliquer sur B1</p> <p>Apply</p>
B1	100 total ACU 1.75 GB memory A-Series compute equivalent 33.18 EUR/Month (Estimated)		
<p>New App Service Plan <input type="checkbox"/> ></p> <p>Create a plan for the web app</p> <p>* App Service plan WebAppA11yDockerPlan </p> <p>* Location West Europe </p> <p>* Pricing tier B1 Basic </p>	<p>Vérifier, puis OK</p>		
<p>* Configure container </p>	<p>Cliquer sur Configure container</p>		
<p>Apply</p>	<p>Single Container, Quickstart</p> <p>Apply</p> <p>(nous reviendrons sur cette configuration ultérieurement)</p>		

	<p>Create</p> <p>Pin to dashboard</p>
	<p>Répéter l'opération pour la Web App « Staging »</p>
	<p>Vérifier, Create</p> <p>Pin to dashboard</p>

WebAppA11yDocker... Web App	Cliquer sur l'une des deux Web App for Container
Running 	Cliquez sur l'URL qui apparaît
Welcome to nginx! If you see this page, the nginx web server is successfully installed and working. Further configuration is required. For online documentation and support please refer to nginx.org . Commercial support is available at nginx.com . Thank you for using nginx.	Une page Web s'ouvre confirmant le bon fonctionnement de la Web App utilisant pour l'instant un container avec un serveur nginx

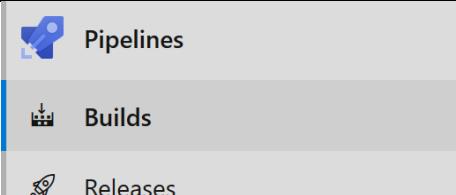
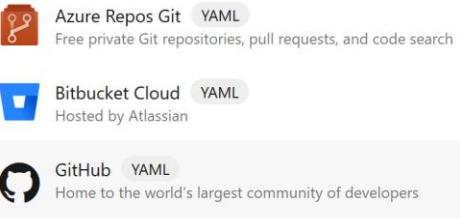
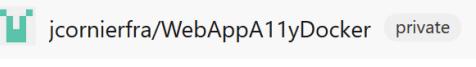
6.5 Création du pipeline DevOps

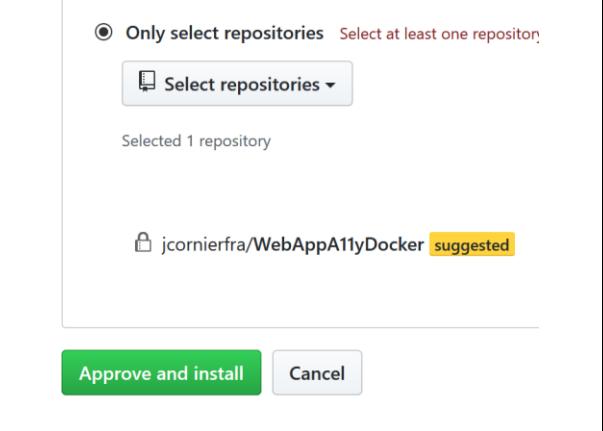
Le principe est le même que pour l'application classique. Les différences sont liées au modèle de container qui implique une intégration et un déploiement adaptés.

6.5.1 Création d'un nouveau Projet DevOps

Illustration	Action
 Azure DevOps My organizations A A11yDemo	Ouvrir https://dev.azure.com et cliquer sur l'organisation
+ Create project	Create project
Project name * <input type="text" value="WebAppA11yDocker"/> Description <div style="border: 1px solid #ccc; padding: 5px; min-height: 40px;">Simple Web App in Docker Container used to demonstrate A11y automatic tests</div> Visibility <div style="display: flex; align-items: center;"> <input type="radio"/> Public <input checked="" type="radio"/> Private <small>Only people you give access to will be able to view this project.</small> </div>	Entrer le nom de la Web App Entrer une description Choisir Private Create

6.5.2 Créeation du pipeline « Builds » DevOps (YAML)

Illustration	Action
	Cliquer sur Pipelines
No build pipelines were found Automate your build in a few easy steps with a new pipeline. New pipeline	New pipeline
Where is your code?  <ul style="list-style-type: none"> Azure Repos Git <small>YAML</small> Free private Git repositories, pull requests, and code search Bitbucket Cloud <small>YAML</small> Hosted by Atlassian Github <small>YAML</small> Home to the world's largest community of developers 	Cliquer sur GitHub YAML
Select a repository <input type="text" value="Filter by keywords"/>  <p>jgormierfra/WebAppA11yDocker <small>private</small></p>	Cliquer sur le référentiel de la Web App WebAbbA11yDocker
Confirm password to continue <div style="border: 1px solid #ccc; padding: 10px; width: fit-content; margin: auto;"> <div style="display: flex; justify-content: space-between; align-items: center;"> Password Forgot password? </div> <div style="border: 1px solid #ccc; height: 30px; margin-top: 5px;"></div> <div style="background-color: green; color: white; padding: 5px 15px; font-weight: bold; text-align: center; margin-top: 10px;">Confirm password</div> </div>	Saisir ou confirmer le mot de passe pour GitHub

	<p>Accéder au bas de la page</p> <p>Sélectionner uniquement le nouveau référentiel</p> <p>Approve and install</p>
	<p>Choisir le compte à utiliser (en principe déjà connecté)</p>
<h3>Configure your pipeline</h3> <p> ASP.NET Core Build and test ASP.NET Core projects targeting .NET Core.</p>	<p>Cliquer sur ASP.NET Core</p>
<p>New pipeline</p> <h3>Review your pipeline YAML</h3> <p>azure-pipelines.yml</p> <pre> 1 # ASP.NET Core 2 # Build and test ASP.NET Core projects targeting .NET Core. 3 # Add steps that run tests, create a NuGet package, deploy, and more: 4 # https://docs.microsoft.com/azure/d 5 6 trigger: 7 - master 8 </pre>	<p>Un script YAML est proposé par défaut, nous devons l'adapter à notre application</p>

6.5.3 Modification du fichier YAML

Remplacer le code par défaut par le code ci-dessous sans valider pour le moment !

```

# ASP.NET Core
# Build and test ASP.NET Core projects targeting .NET Core.
# Add steps that run tests, create a NuGet package, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core

trigger:
- master

```

```

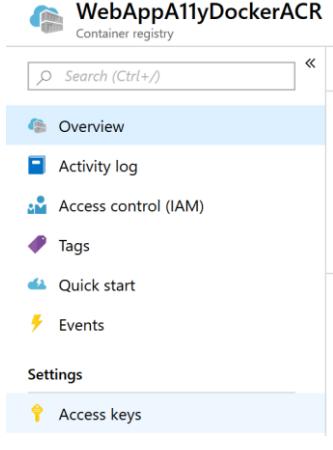
pool:
  vmImage: 'Ubuntu 16.04'

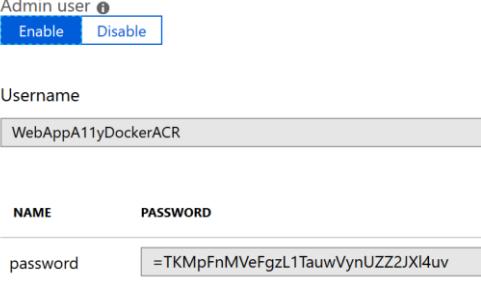
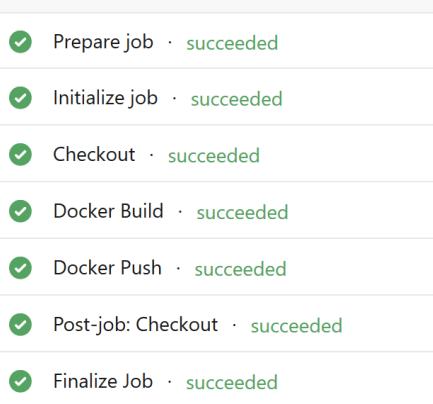
variables:
  imageName: 'webappa11ydocker:${Build.BuildId}'
  dockerId: ' webappa11ydockeracr'
  dockerPassword: 'bR4vMqhs9ii8iU/JCdLk3u8BmsKWksE'

steps:
- script: docker build -t $(dockerId).azurecr.io/$imageName .
  workingDirectory: WebAppA11yDocker
  displayName: 'Docker Build'

- script: |
    docker login -u $(dockerId) -p $pswd $(dockerId).azurecr.io
    docker push $(dockerId).azurecr.io/$imageName
  env:
    pswd: $(dockerPassword)
  workingDirectory: /
  displayName: 'Docker Push'

```

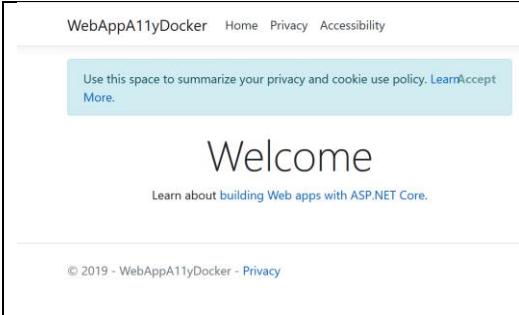
Portail Azure	
WebAppA11yDocker... RegistryResource 	Ouvrir https://portal.azure.com Accéder au référentiel de container (cliquer sur l'icône sur le dashboard ou aller dans Container registries)
	Cliquer sur Access keys

	<p>Cliquer sur Enable</p> <p>Copier le password</p>
Azure DevOps	
<pre>variables: - imageName: 'webappa11ydocker:\${Build.BuildId}' - dockerId: 'webappa11ydocker.acr' - dockerPassword: 'Q6Y+IXgYHrRY0IEsSbnY6wAz=KXYRtl8'</pre>	<p>Retourner sur la page du pipeline DevOps</p>
	<p>Remplacer le password par le nouveau</p> <p>Save and run</p>
Job Job Pool: Hosted Ubuntu 1604 · Agent: Hosted Agent 	<p>Les tâches de construction doivent réussir</p>

6.5.4 Configuration des Web App for Containers Azure

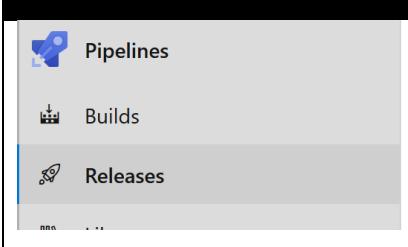
Illustration	Action
	<p>Ouvrir le portail Azure</p> <p>https://portal.azure.com</p> <p>Accéder au référentiel de containers</p>

Services	Cliquer sur repositories
Repositories	
REPOSITORIES webappa11ydocker	L'image que nous avons préalablement construite et publiée devrait apparaître
WebAppA11yDocker... Web App Running	Revenir sur le dashboard Cliquer sur la Web App Docker Staging
Settings Configuration Container settings	Cliquer sur Container settings
 Image source Azure Container Registry <input checked="" type="radio"/> Docker Hub <input type="radio"/> Private Registry Registry WebAppA11yDockerACR Image webappa11ydocker Tag 142 Startup File Continuous Deployment On <input checked="" type="radio"/> Off Webhook URL show url	Cliquer sur Azure Container Registry Sélectionner le nom du référentiel Sélectionner l'image Sélectionner le tag avec le n° le plus élevé Save
WebAppA11yDockerStaging App Service Search (Ctrl+/ Overview	Cliquer sur overview
URL : https://webappa11ydockerstaging.azuredockerapp.com	Cliquer sur l'URL

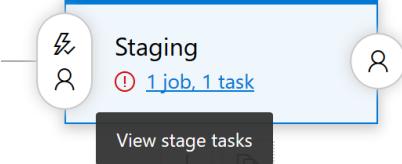
	<p>Une page Web s'ouvre</p> <p>Patienter le temps du redémarrage du container</p> <p>Le site web devrait apparaître</p>
	<p>Réitérer les opérations pour la Web App for Container de production</p>

6.5.5 Créeation du pipeline « Release » du Container

La création du pipeline « Release » est similaire à celle de l’application classique à l’exception du choix des tâches de déploiement du Service d’Application qui cette fois doit être pour container Linux.

Illustration	Action
	<p>Sélectionner Releases</p>
<p>No release pipelines found</p> <p>Automate your release process in a few easy steps with a new pipeline</p> <p>New pipeline</p>	<p>New pipeline</p>
<p>Featured</p> <p>Azure App Service deployment</p> <p>Deploy your application to Azure App Service. Choose from Web App on Windows, Linux, containers, Function Apps, or WebJobs.</p>	<p>Choisir Azure App Service deployment</p> <p>Apply</p>

<p>Stage</p> <p>Staging</p> <p> Properties ^ Name and owners of the stage</p> <p>Stage name <u>Staging</u></p> <p>Stage owner Jerome Cornier</p>	<p>Renommer en « Staging »</p>
<p>Add an artifact</p> <p> Schedule not set</p>	<p>Cliquer sur Add an artifact</p>
<p>Add an artifact</p> <p>Source type Build Azure Repos ... GitHub 4 more artifact types ▾ </p> <p>Project * <input type="text" value="WebAppA11yDocker"/></p> <p>Source (build pipeline) * <input type="text" value="WebAppA11yDocker-ASP.NET-CI"/></p> <p>Default version * <input type="text" value="Latest"/></p> <p>Source alias * <input type="text" value="_WebAppA11yDocker-ASP.NET-CI"/></p> <p> No version is available for WebAppA11yDocker-ASP.NET-CI publish. Please check the source pipeline.</p> <p>Add</p>	<p>Sélectionner le projet (Docker)</p> <p>Sélectionner la source : le pipeline créé précédemment</p> <p>Les autres champs sont automatiquement renseignés</p> <p>Add</p>
<p>_WebAppA11yDocker-ASP.NET-CI</p>	<p>Cliquer sur l'icône de l'éclair dans le coin de la boîte de l'Artifact</p>

<p>Continuous deployment trigger</p> <p>Build: _WebAppA11yDocker-ASP.NET-CI</p> <p><input checked="" type="checkbox"/> Enabled Creates a release every time a new build is available.</p> <p>Build branch filters (i)</p>	<p>Cliquer sur l'interrupteur pour activer le déclenchement automatique du pipeline Release</p>
	<p>Cliquer sur 1 job, 1 task</p>
<p>Stage name</p> <input type="text" value="Staging"/> <p>Parameters (i) Unlink all</p> <p>Azure subscription * Manage ↗</p> <p>Visual Studio Ultimate with MSDN (a064ddb8-d1)</p> <p>Scoped to subscription 'Visual Studio Ultimate with MSDN'</p> <p>App type</p> <input type="text" value="Web App for Containers (Linux)"/> <p>App service name *</p> <input type="text" value="WebAppA11yDockerStaging"/> <p>Registry or Namespace *</p> <input type="text" value="webappa11ydockeracr.azurecr.io"/> <p>Repository *</p> <input type="text" value="webappa11ydocker"/>	<p>Les principales différences dans la version en Container se trouvent ici !</p> <p>Sélectionner la souscription Azure</p> <p>Sélectionner Web App for Containers (Linux)</p> <p>Sélectionner la Web App "Staging"</p> <p>Registry : saisir le nom complet du référentiel (webappa11ydockeracr.azurecr.io)</p> <p>Repository : saisir le nom de l'application (sans acr à la fin)</p>
<p>↑ New release pipeline </p>	<p>Positionner la souris sur New release pipeline</p> <p>Cliquer sur le stylo</p>
<p>↑ <u>WebAppA11yDocker-ASP.NET-CD</u></p>	<p>Changer le nom du pipeline</p>
<p> Save </p>	<p>Save</p> <p>OK</p>

	Cliquer sur All pipelines
	Cliquer sur le nom du pipeline
	Create release Create
	Cliquer sur le nom de la Release en cours
	Patienter
	Le déploiement devrait être réussi

6.6 Implémentation des tests d'accessibilité automatisés

L'implémentation des tests d'accessibilité est identique à celle mise en œuvre pour le prototype précédent, avec exactement les mêmes résultats en termes de conformité puisque l'application source est la même, présentée également sous forme de site Web à l'agent de test d'accessibilité, seule l'architecture en container diffère.

Les étapes de l'implémentation de l'Agent sont identiques.

La configuration de la Web App for Containers de production se configure comme la Web App Staging que nous venons de décrire.

6.7 Pipeline final

Voici à quoi ressemble le pipeline CI/CD complet.

6.7.1 CI

Job Job		Started: 17/04/2019 à 22:15:11
Agent:		
Pool: Hosted Ubuntu 1604	· Hosted Agent	··· 2m 37s
✓ Prepare job	· succeeded	<1s
✓ Initialize job	· succeeded	<1s
✓ Checkout	· succeeded	8s
✓ Docker Build	· succeeded	2m 23s
✓ Docker Push	· succeeded	3s
✓ Post-job: Checkout	· succeeded	<1s
✓ Finalize Job	· succeeded	<1s

6.7.2 CD

