Computer Training
by Netmind

Boosting
Your
Digital
Skills

MI1575

Taller DevOps

Web App Node - Githib integration - Pull requests

Taller DevOps

Overview

In this lab, you will learn how you can connect Azure with Github, add testing and production stages to the Pipelines to deploy a Node web application

Fork the Node & Express Demo App Repository

The Node and Express app is a simple website for a fictitious company. This app uses Express and Handlebars to serve up a few common pages you'd see on any company website. Also included are some unit tests that ensure those routes are working and serving up the right content.

You can head over to my GitHub account to fork this repository.

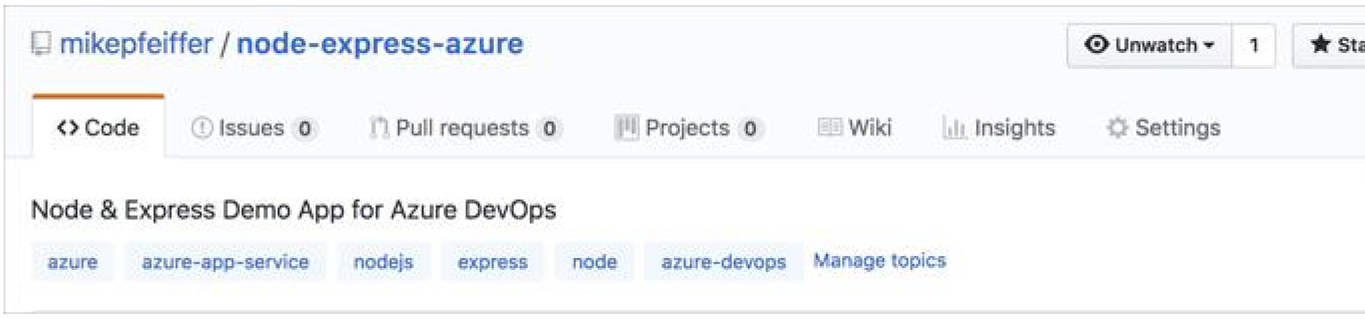


Figure 3. Fork the node-express-azure Repository.

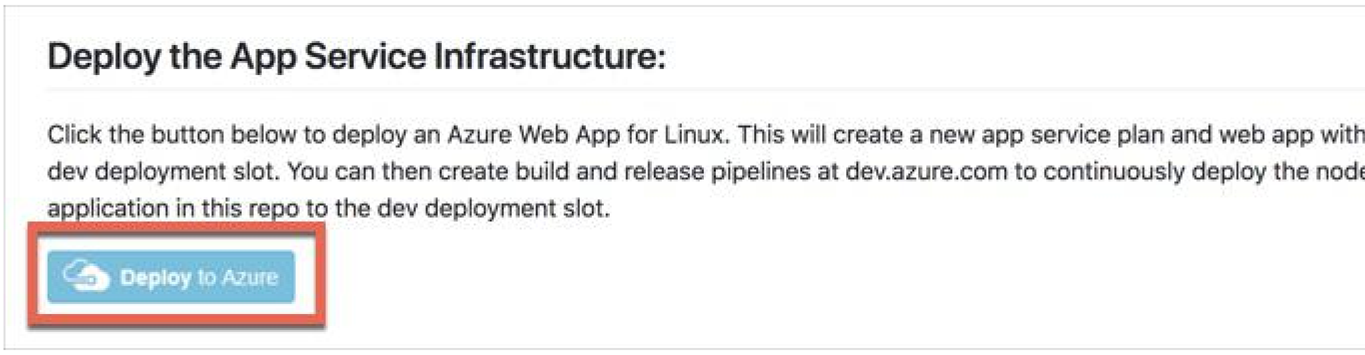
Next, we can move on to deploying the infrastructure to support both development and production deployment slots using Azure App Service.

Deploy the App Service Infrastructure

We're going to use an Azure Web App for Linux resource to power our Node and Express application. We'll set things up so our CI/CD pipeline can build and deploy the app into a development/staging slot. Then we'll set up up a manual approval into the production slot.

We'll use an Azure Resource Manager (ARM) template to build the App Service infrastructure.

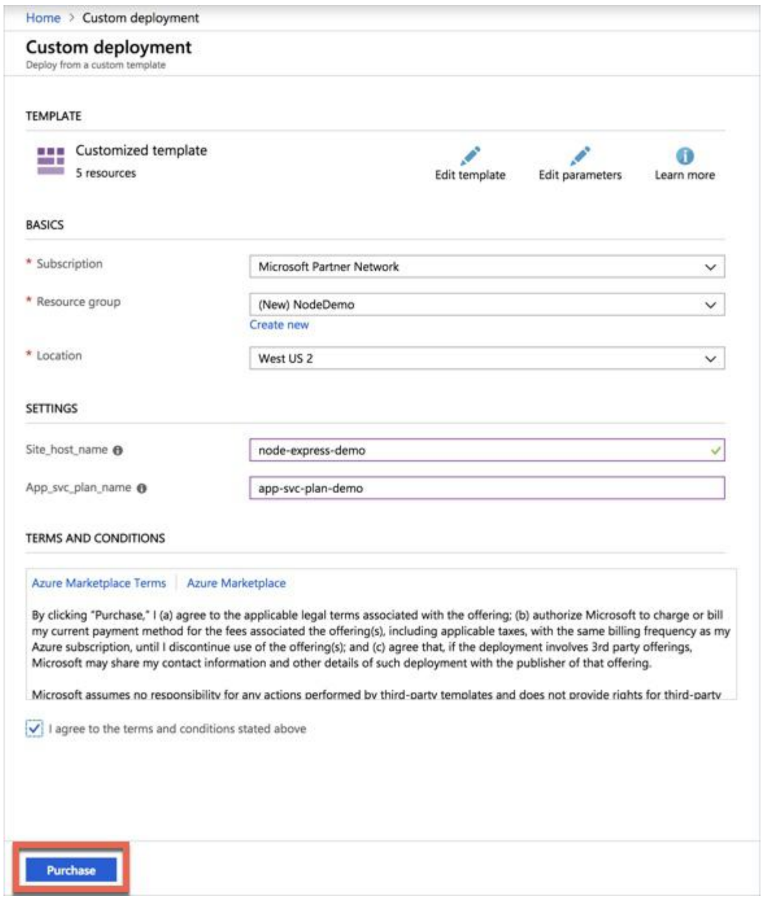
Navigate to the node-express-azure repository you forked in the previous step. You'll see a "Deploy to Azure" button about halfway down the screen.


Deploy the App Service Infrastructure:

Click the button below to deploy an Azure Web App for Linux. This will create a new app service plan and web app with
dev deployment slot. You can then create build and release pipelines at dev.azure.com to continuously deploy the nod
application in this repo to the dev deployment slot.

Figure 4. Deploying the ARM Template.

Clicking the "Deploy to Azure" button will redirect you to the Azure portal as shown in Figure 5.



* Subscription	Microsoft Partner Network
* Resource group	(New) NodeDemo
* Location	West US 2
Site_host_name	node-express-demo
App_svc_plan_name	app-svc-plan-demo

By clicking "Purchase," I (a) agree to the applicable legal terms associated with the offering; (b) authorize Microsoft to charge or bill my current payment method for the fees associated the offering(s), including applicable taxes, with the same billing frequency as my Azure subscription, until I discontinue use of the offering(s); and (c) agree that, if the deployment involves 3rd party offerings, Microsoft may share my contact information and other details of such deployment with the publisher of that offering.

Microsoft assumes no responsibility for any actions performed by third-party templates and does not provide rights for third-party

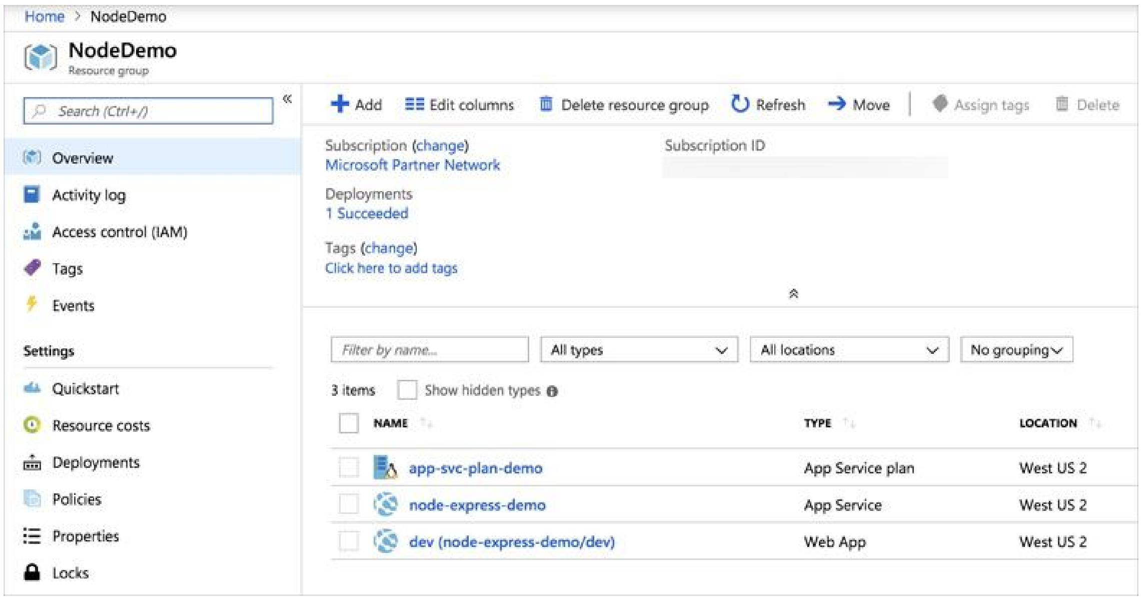
☑ I agree to the terms and conditions stated above

Figure 5. Launching the ARM Template.

Notice that you'll need to set a globally unique hostname for your web application, along with a name for the new app service plan. I'd recommend deploying these resources into a new resource group. That way when you're done with this walkthrough, you can clean up the Azure resources easily by deleting the resource group.

Click "Purchase" to launch the template to agree that you'll have to pay for the App Service resources that this template deploys on your behalf.

After you launch the template you should see a successful deployment message, and you should have a new resource group similar to the one shown in Figure 6. Notice that there is an App Service Plan, a web app that represents the production deployment slot, and a slot for development called "dev".



NodeDemo
Resource group

Subscription (change)
Microsoft Partner Network

Subscription ID

Deployments
1 Succeeded

Tags (change)
Click here to add tags

3 items

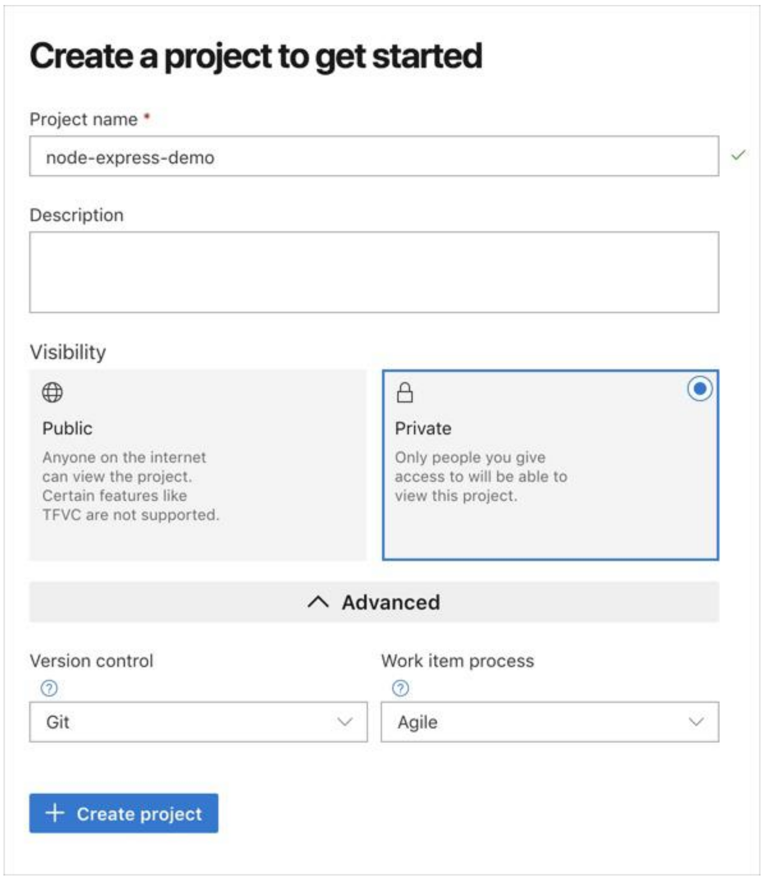
NAME	TYPE	LOCATION
app-svc-plan-demo	App Service plan	West US 2
node-express-demo	App Service	West US 2
dev (node-express-demo/dev)	Web App	West US 2

Figure 6. Reviewing the Resources.

Quick side note about the ARM template: the Deploy to Azure button references the azuredeploy.json ARM template in my GitHub repository. If you want to update the template, update the version in your own repo, and don't forget to change the target of the button in the source of your README.md file.

Create a Build Pipeline

We're ready to move on and set up a build pipeline in Azure DevOps. Head back to dev.azure.com and create a new project inside your organization. Use the settings shown in Figure 7.



Create a project to get started

Project name *

node-express-demo

Description

Visibility

Public
Anyone on the internet can view the project. Certain features like TFVC are not supported.

[✓ Private]
Only people you give access to will be able to view this project.

Advanced

Version control: Git

Work item process: Agile

Figure 7. The New Project Settings.

After clicking on the “Create project” button, you’ll see a summary page for the project. Navigate to Pipelines and click on Builds as shown in Figure 8.

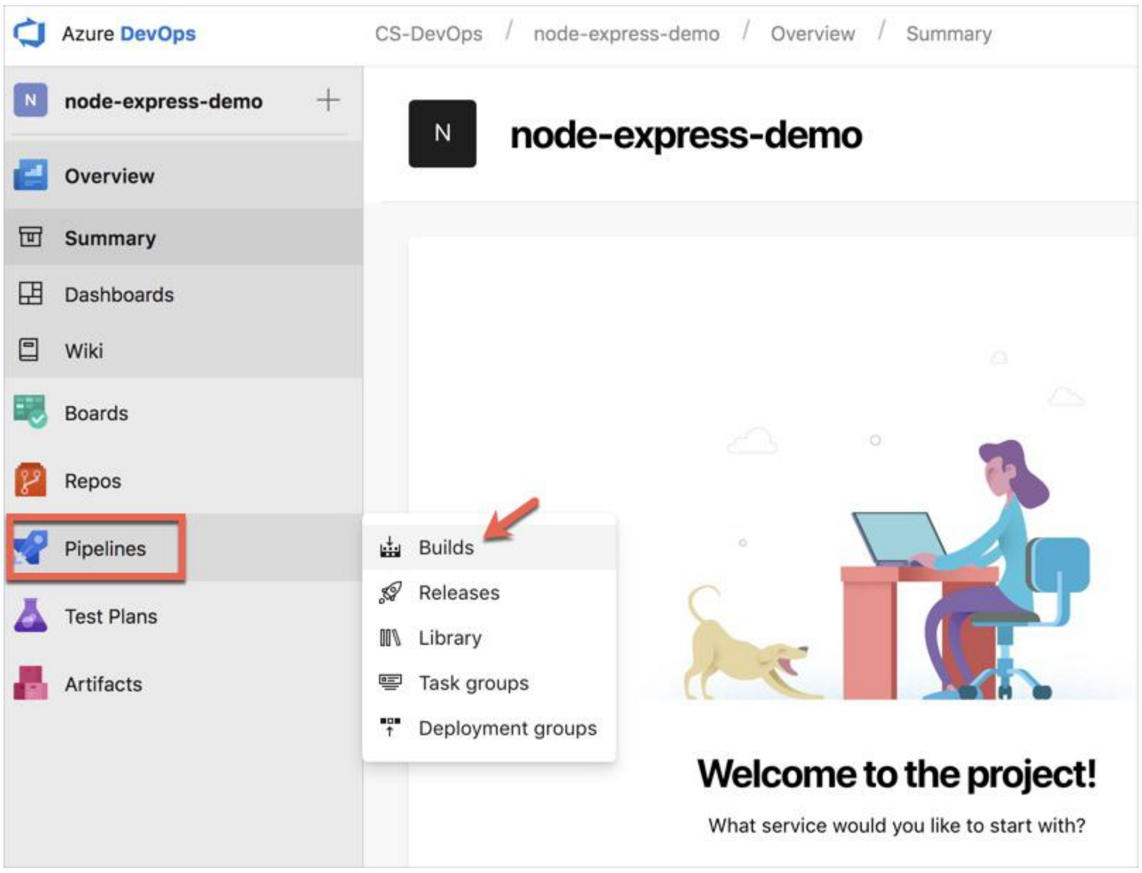
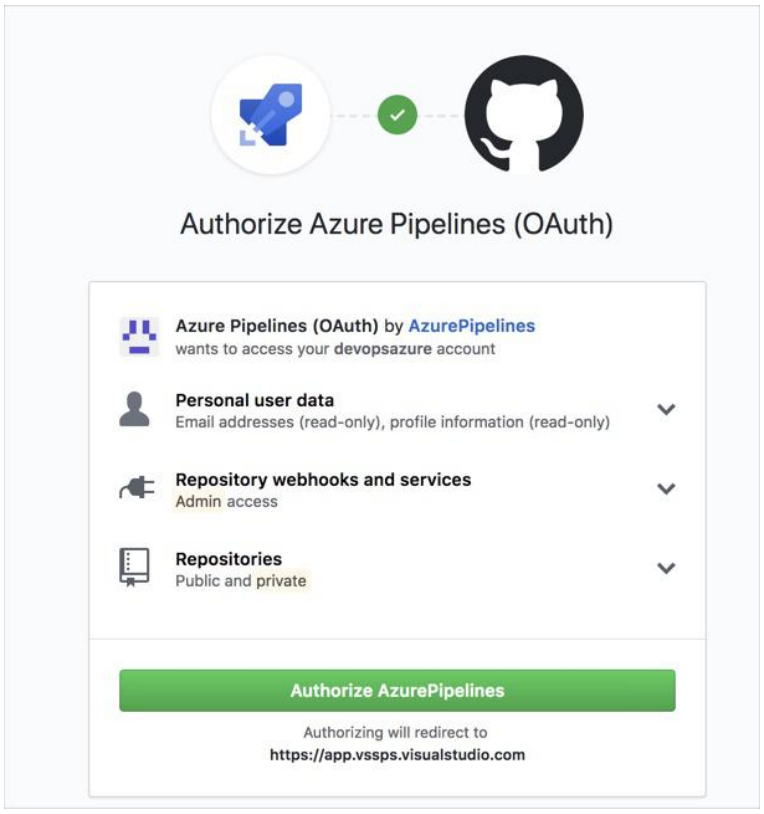


Figure 8. Creating a New Build Pipeline.

Next, click the button to create a new build pipeline. You'll be prompted to choose a repository. Select GitHub. You'll see a screen like the one in Figure 9 where you'll need to authorize the Azure DevOps service to connect to your GitHub account on your behalf. Click Authorize.


Authorize Azure Pipelines (OAuth)

Azure Pipelines (OAuth) by AzurePipelines
wants to access your devopsazure account

Personal user data
Email addresses (read-only), profile information (read-only)

Repository webhooks and services
Admin access

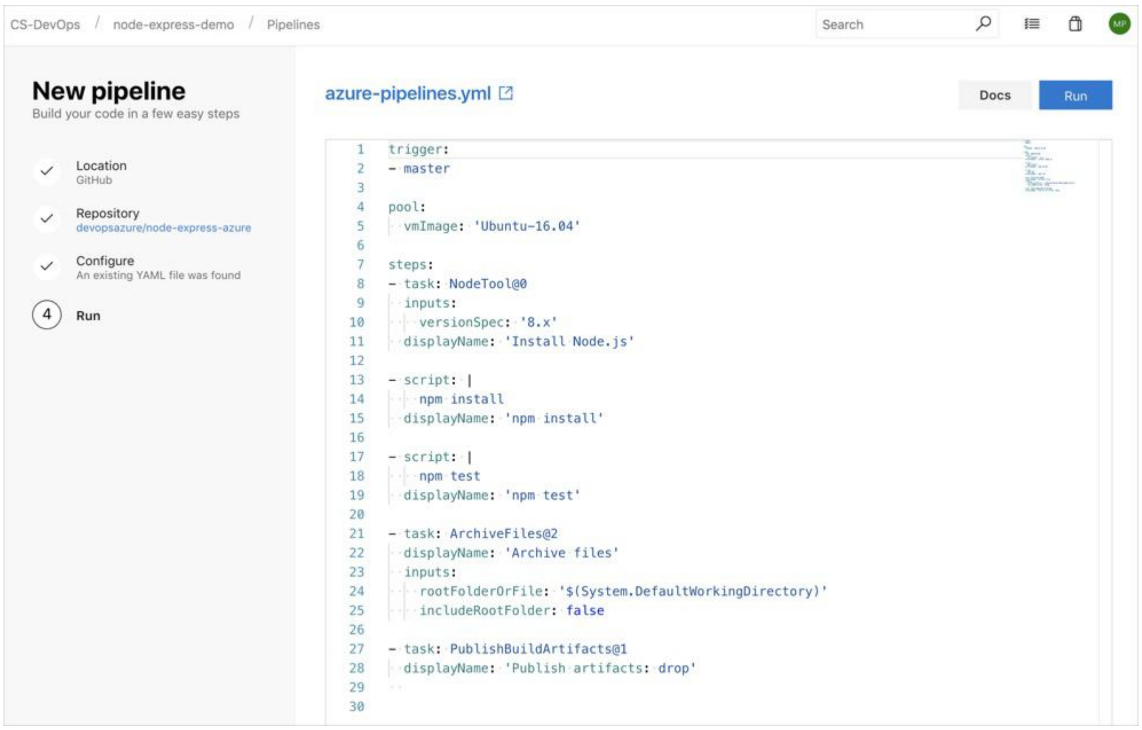
Repositories
Public and private

Authorize AzurePipelines

Authorizing will redirect to
https://app.vssps.visualstudio.com

Figure 9. Authorizing the GitHub Connection.

After your connection to GitHub has been authorized select the node-express-azure repo that you forked in the first step. You should end up seeing a "New pipeline" screen like the one shown in Figure 10.



New pipeline
Build your code in a few easy steps

- Location: GitHub
- Repository: devopsazure/node-express-azure
- Configure: An existing YAML file was found
- 4 Run

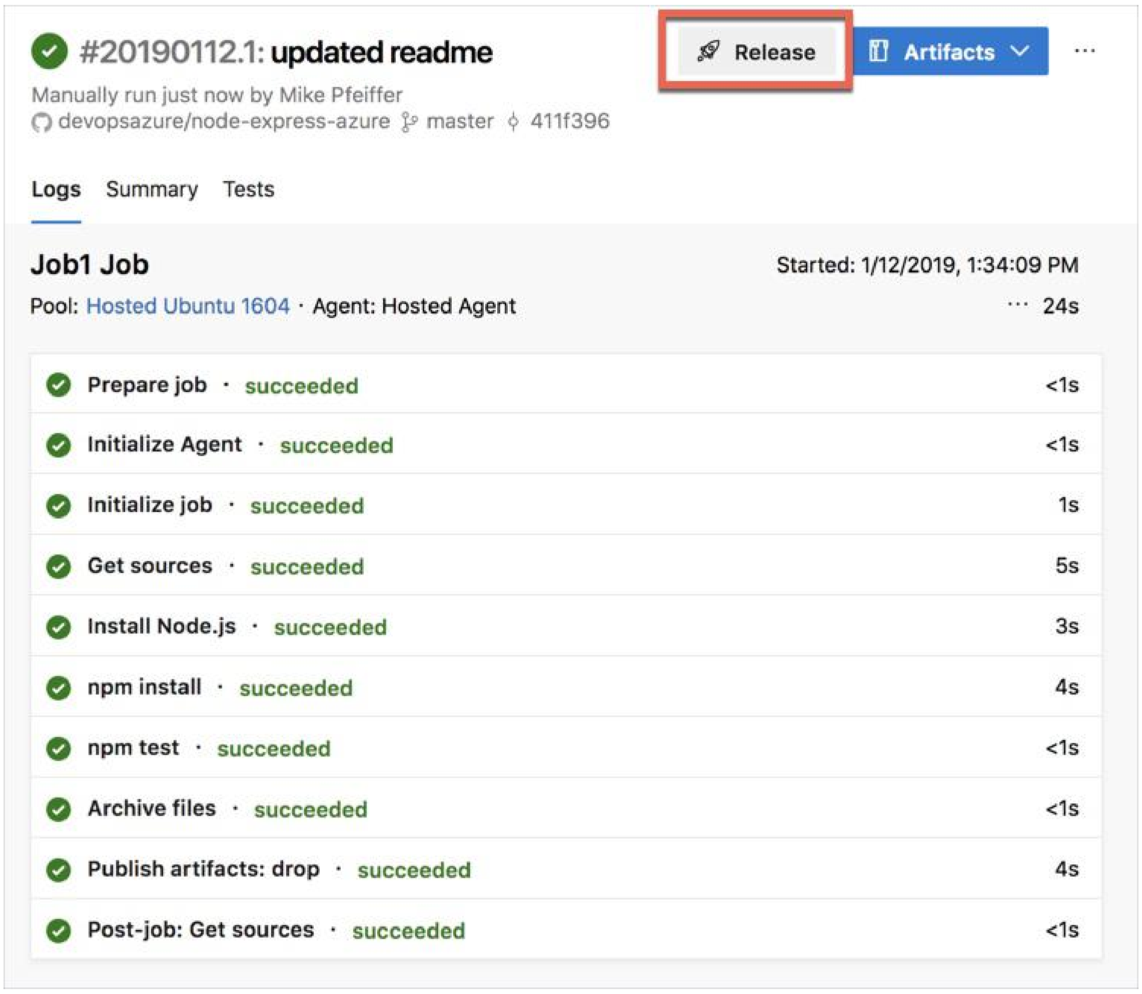
azure-pipelines.yml

```yaml
trigger:
- master

pool:
  vmImage: 'Ubuntu-16.04'

steps:
- task: NodeTool@0
  inputs:
    versionSpec: '8.x'
  displayName: 'Install Node.js'

- script: |
    npm install
  displayName: 'npm install'

- script: |
    npm test
  displayName: 'npm test'

- task: ArchiveFiles@2
  displayName: 'Archive files'
  inputs:
    rootFolderOrFile: '$(System.DefaultWorkingDirectory)'
    includeRootFolder: false

- task: PublishBuildArtifacts@1
  displayName: 'Publish artifacts: drop'
```

Figure 10. Creating the New Build Pipeline.

The new pipeline wizard should recognize that we already have an azure-pipelines.yml in the repository. This file contains all of the settings that the build service should use to build and test our application, as well as generate the output artifacts that will be used to deploy the app later in our release pipeline.

After you click "Run" to kick off your first build, you should see a screen like the one shown in Figure 11.



#20190112.1: updated readme
Manually run just now by Mike Pfeiffer
devopsazure/node-express-azure master 411f396

Job1 Job
Started: 1/12/2019, 1:34:09 PM
Pool: Hosted Ubuntu 1604 · Agent: Hosted Agent ··· 24s

```
Prepare job · succeeded                 <1s
Initialize Agent · succeeded            <1s
Initialize job · succeeded               1s
Get sources · succeeded                  5s
Install Node.js · succeeded              3s
npm install · succeeded                  4s
npm test · succeeded                    <1s
Archive files · succeeded               <1s
Publish artifacts: drop · succeeded      4s
Post-job: Get sources · succeeded       <1s
```

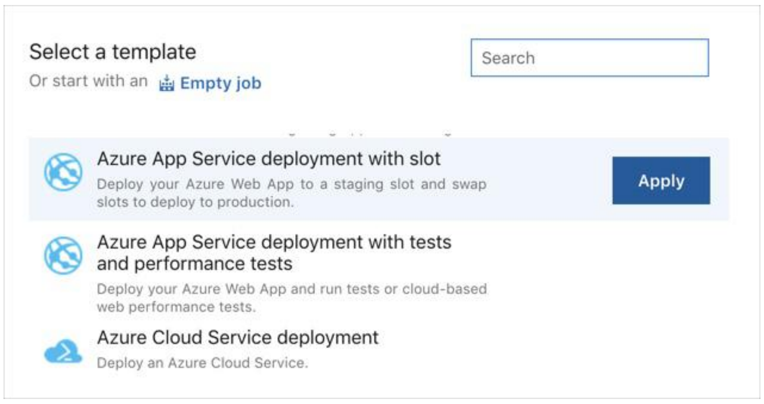
Figure 11. Reviewing the Build Status.

Notice that a lot went on with the build. The service used an Ubuntu 16.04 build agent to grab the code from GitHub, installed our development dependencies, and then ran our unit tests to validate the application. Finally, the code was bundled into an output artifact and published so we can use it as an input artifact for our upcoming release pipeline.

Click on the release button at the top of this screen to create a new release pipeline.

Create a Release Pipeline

When you get into the release pipeline screen, you'll need to select a template. For this scenario, we are going to choose "App Service deployment with slot".


Select a template

Or start with an Empty job

Search

Azure App Service deployment with slot

Deploy your Azure Web App to a staging slot and swap slots to deploy to production.

Apply

Azure App Service deployment with tests and performance tests

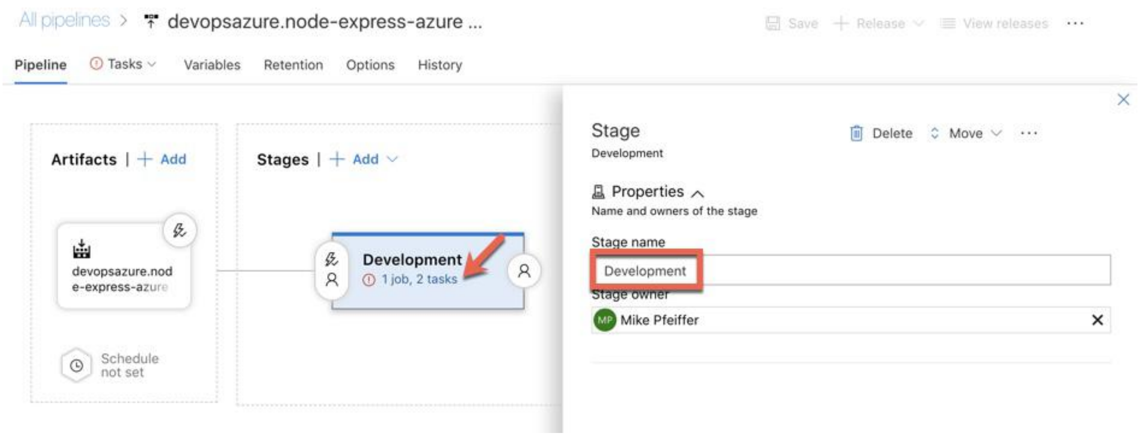
Deploy your Azure Web App and run tests or cloud-based web performance tests.

Azure Cloud Service deployment

Deploy an Azure Cloud Service.

Figure 12. Selecting the Deployment Template.

Click on the apply button to create the new deployment stage within the release pipeline. On the next screen, you'll be able to configure this stage. Change the name to "development" as shown in Figure 13.



Stage

Development

Properties

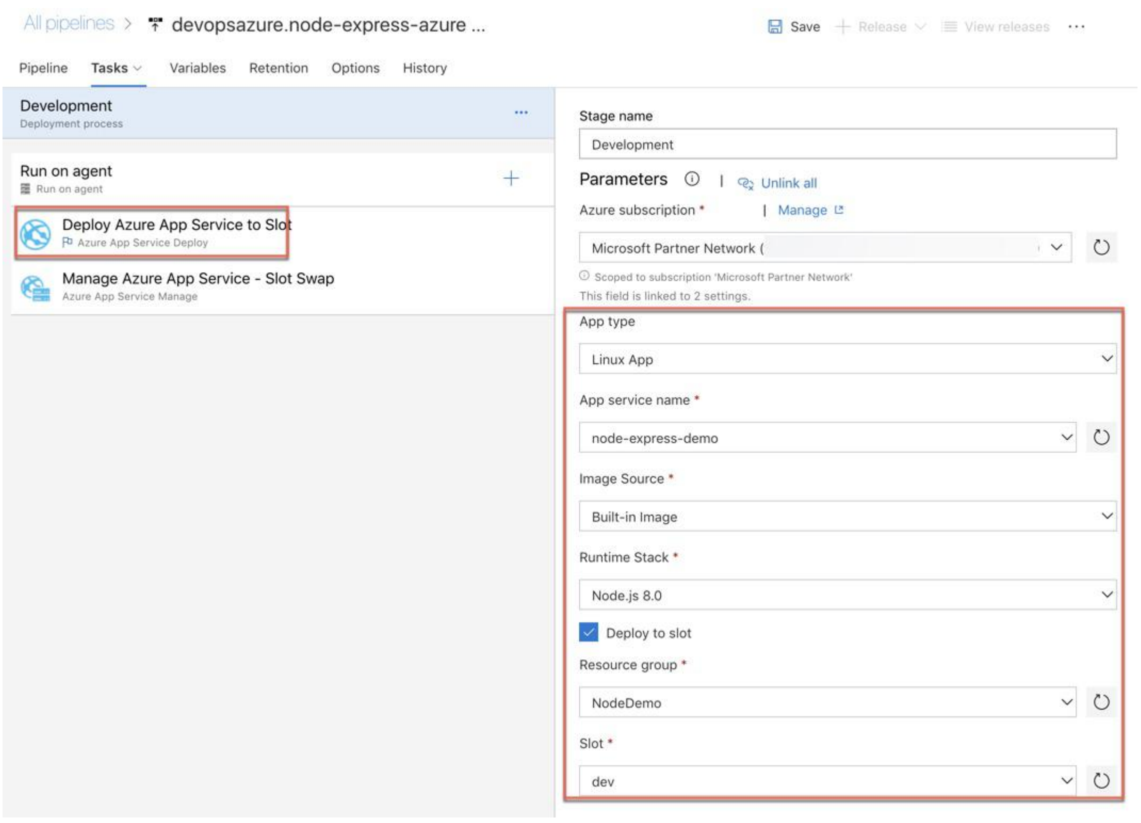
Name and owners of the stage

Stage name: Development

Stage owner: Mike Pfeiffer

Figure 13. Updating the Development Stage.

While on this screen, click on the link that says "2 tasks" inside your development stage. This will take you to a screen where you can configure the deployment task. Make sure you fill out all the fields as shown in Figure 14.


All pipelines > devopsazure.node-express-azure ...

Save | Release | View releases

Pipeline | Tasks | Variables | Retention | Options | History

Development
Deployment process

Run on agent
Run on agent

Deploy Azure App Service to Slot
Azure App Service Deploy

Manage Azure App Service - Slot Swap
Azure App Service Manage

Stage name: Development

Parameters | Unlink all

Azure subscription * | Manage

Microsoft Partner Network (

Scoped to subscription 'Microsoft Partner Network'
This field is linked to 2 settings.

App type: Linux App

App service name *: node-express-demo

Image Source *: Built-in Image

Runtime Stack *: Node.js 8.0

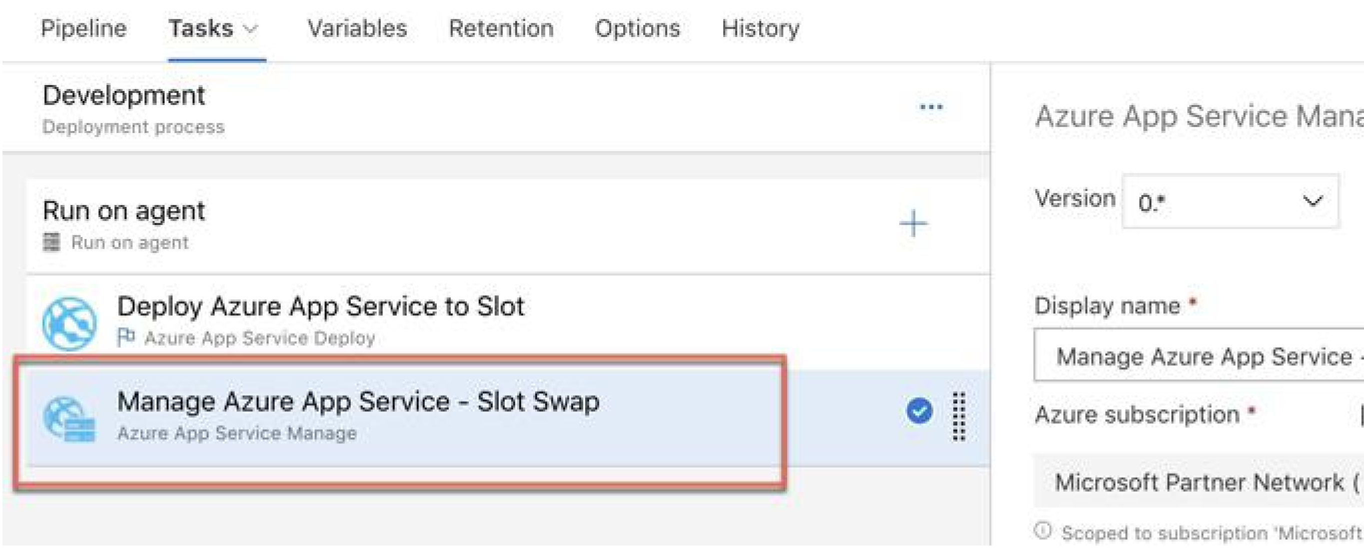
✓ Deploy to slot

Resource group *: NodeDemo

Slot *: dev

Figure 14. Configuring the Deployment Task.

Next, highlight and remove the second deployment task for swapping the slots.


Pipeline | Tasks | Variables | Retention | Options | History

Development
Deployment process

Run on agent
Run on agent

Deploy Azure App Service to Slot
Azure App Service Deploy

Manage Azure App Service - Slot Swap
Azure App Service Manage

Azure App Service Mana

Version: 0.*

Display name *: Manage Azure App Service

Azure subscription *

Microsoft Partner Network (

Scoped to subscription 'Microsoft

Figure 15. Removing the Slot Swap Task.

Finally, click Save.

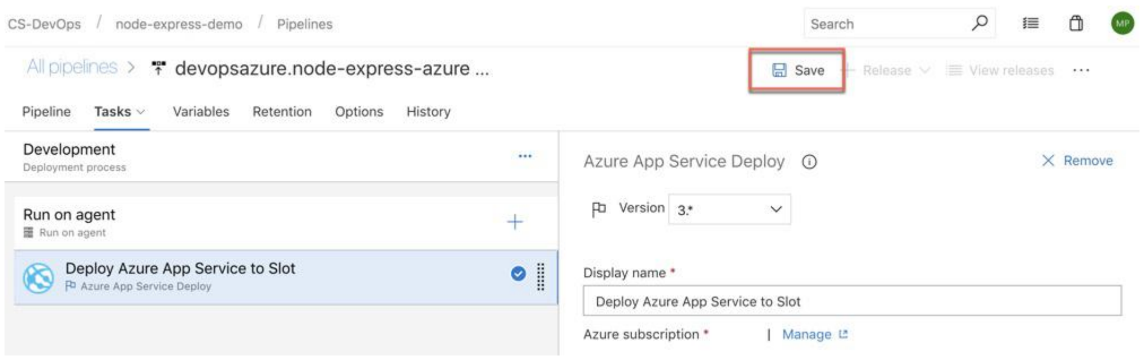


Figure 16. Save the Changes to the Development Stage.

Head back over to the "Pipeline" tab at the top left of the screen. Inspect the deployment triggers for the artifacts as shown in Figure 17.

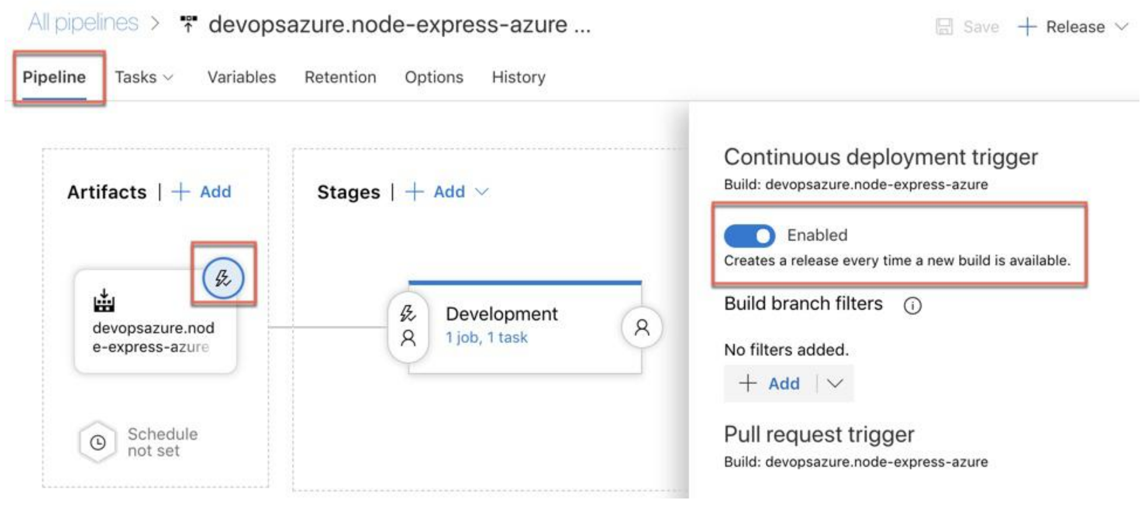

All pipelines > devopsazure.node-express-azure ...
Save
Release
Pipeline
Tasks
Variables
Retention
Options
History
Artifacts | Add
Stages | Add
devopsazure.node-express-azure
Schedule not set
Development
1 job, 1 task
Continuous deployment trigger
Build: devopsazure.node-express-azure
Enabled
Creates a release every time a new build is available.
Build branch filters
No filters added.
Add
Pull request trigger
Build: devopsazure.node-express-azure

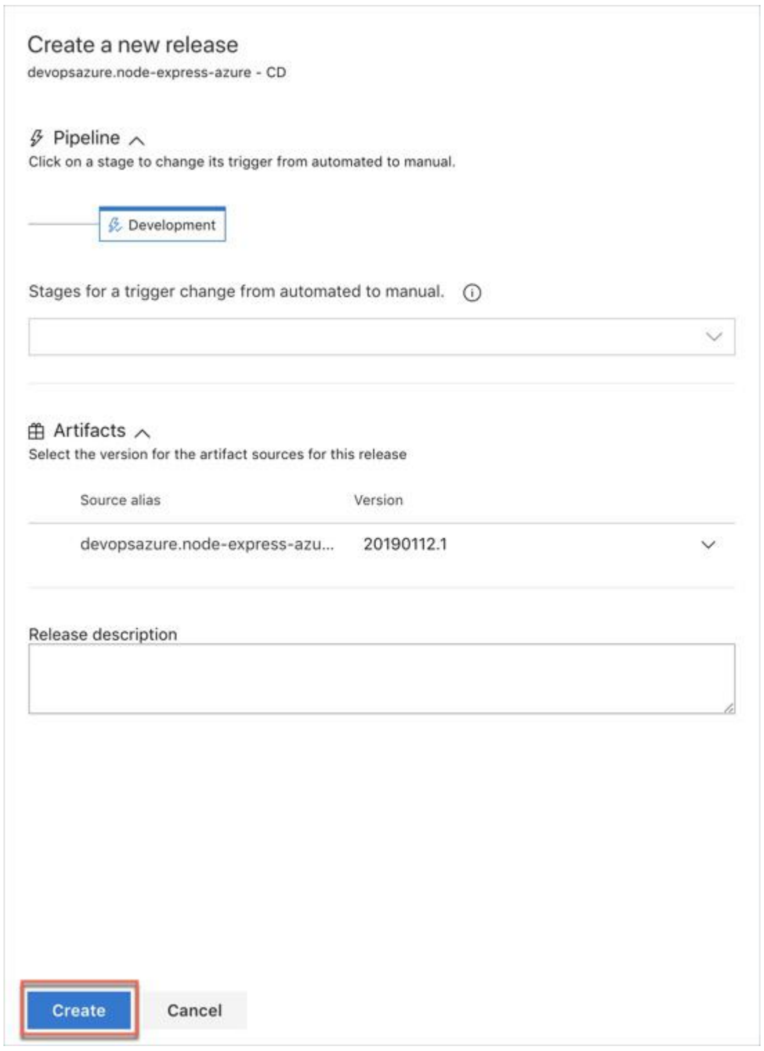
Figure 17. Reviewing the Deployment Trigger.

Notice that continuous deployment is enabled by default. Going forward, each new build will trigger a deployment to our development slot in Azure App Service.

First, let's trigger a manual release.

Create Your First Release to Development

Click the "Release" button on the top right of the release pipeline screen and create a new release. Use the settings as shown in Figure 18.


Create a new release

devopsazure.node-express-azure - CD

Pipeline

Click on a stage to change its trigger from automated to manual.

Stages for a trigger change from automated to manual.

Artifacts

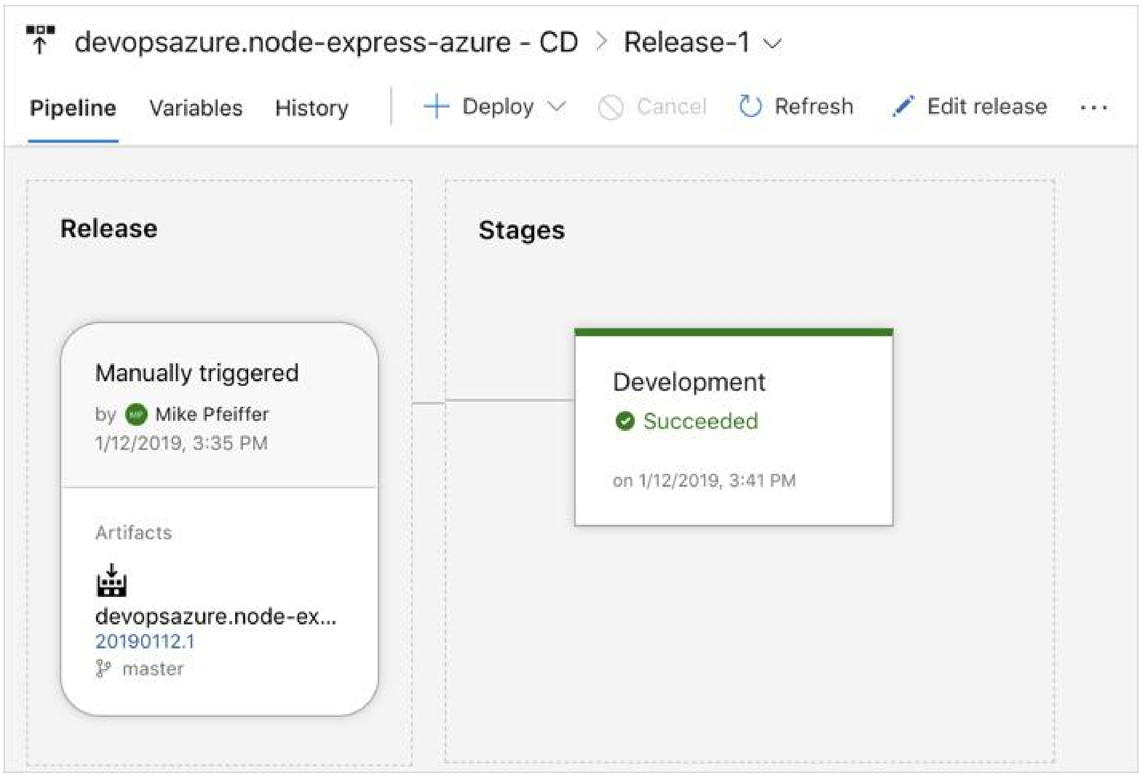
Select the version for the artifact sources for this release

Source alias	Version
devopsazure.node-express-azu...	20190112.1

Release description

Figure 18. Triggering a Release.

Click on the "Create" button to deploy the application to the development deployment slot. You should see a successful status in the properties of the release.


devopsazure.node-express-azure - CD > Release-1

Release

Manually triggered
by Mike Pfeiffer
1/12/2019, 3:35 PM

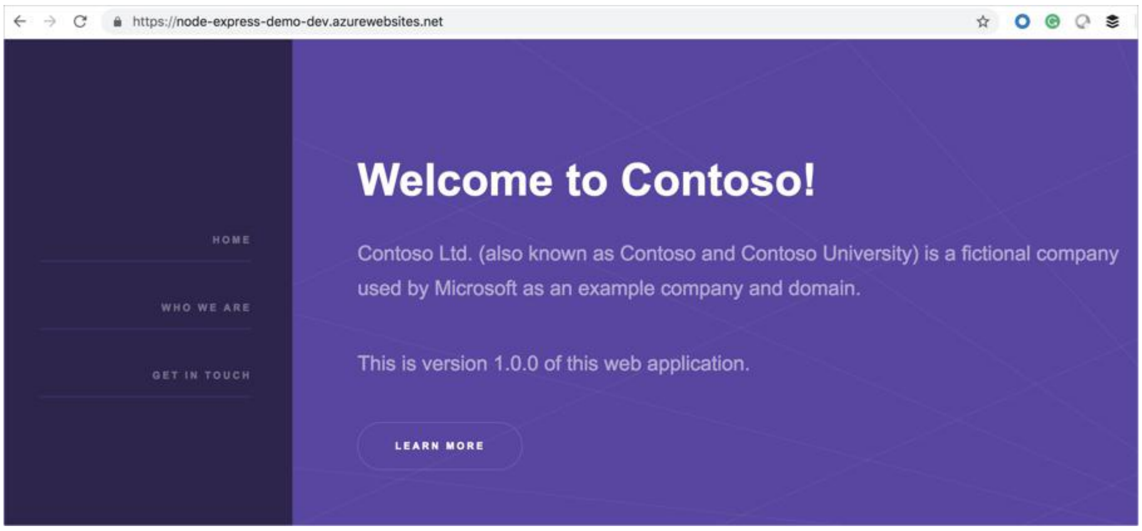
Artifacts
devopsazure.node-ex...
20190112.1
master

Stages

Development
Succeeded
on 1/12/2019, 3:41 PM

Figure 19. Reviewing the Release Status.

Navigate to the public URL of the "dev" deployment slot in your web browser. The hostname will have "-dev" appended to it. For example, my web app is named "node-express-demo" and the "dev" deployment slot URL is https://node-express-demo-dev.azurewebsites.net.


https://node-express-demo-dev.azurewebsites.net

Welcome to Contoso!

Contoso Ltd. (also known as Contoso and Contoso University) is a fictional company used by Microsoft as an example company and domain.

This is version 1.0.0 of this web application.

Figure 20. The Demo Node App Running in the "dev" Deployment Slot.

You should see the sample web application when you visit the "dev" slot URL. The production slot will show the default Azure App Service splash page since it is virtually untouched at this point. Let's change that in the next step.

Create a Release Stage for Production

Head back over to the Azure DevOps portal and go to Pipelines > Releases. Click on the "Edit" button to modify the pipeline. Highlight the Development stage and click the dropdown to clone the stage.

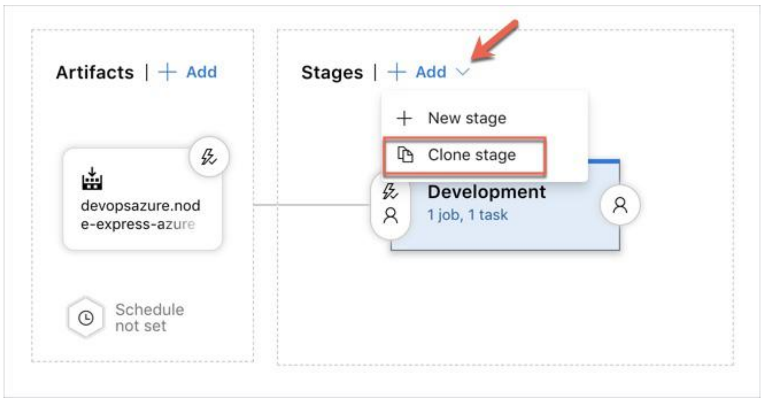
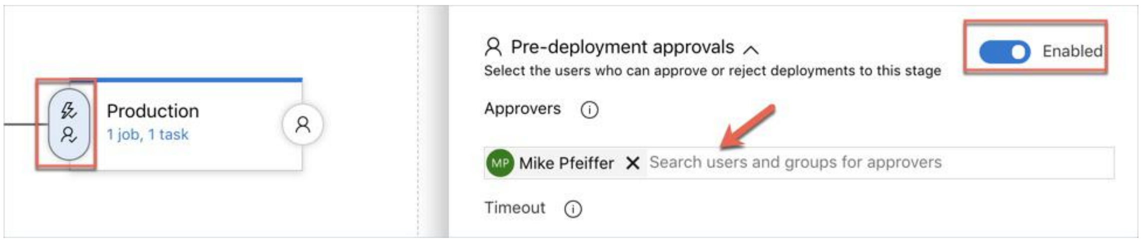


Figure 21. Cloning the Development Stage.

Rename the stage to "Production".

Next, click the pre-deployment conditions button for the Production stage. Enable pre-deployment approvals and add yourself as an approver.



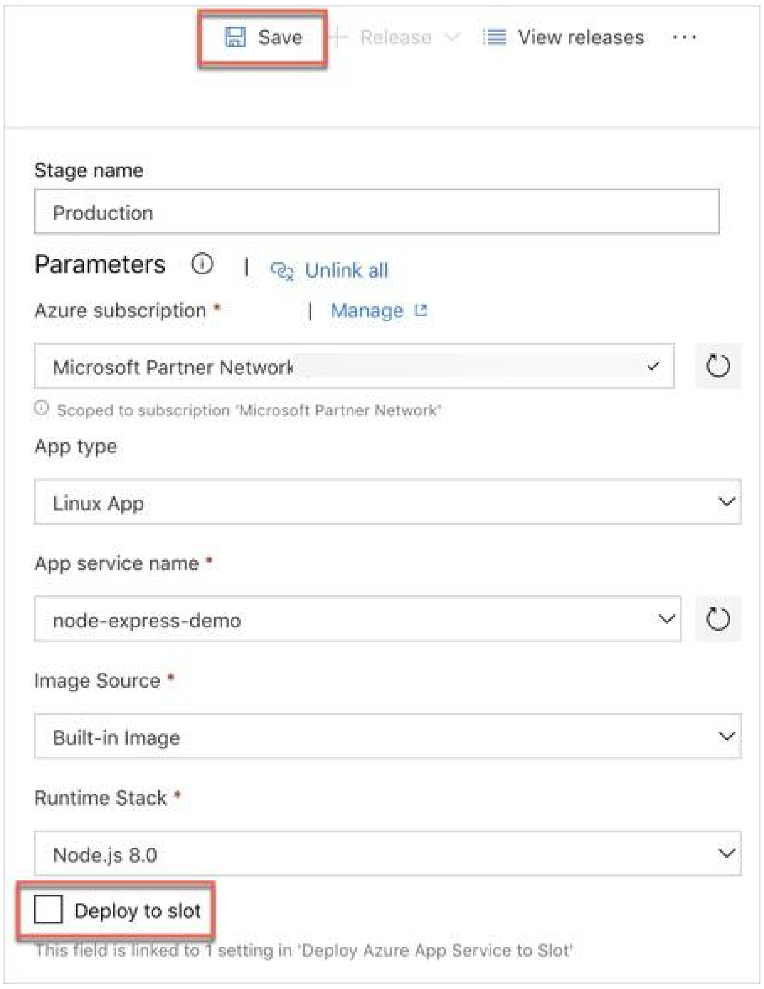
Production
1 job, 1 task
Pre-deployment approvals
Select the users who can approve or reject deployments to this stage
Enabled
Approvers
Mike Pfeiffer
Search users and groups for approvers
Timeout

Figure 22. Requiring Approval for Production Deployment.

We're doing this because we don't want automated deployments going straight into production. We're not building a continuous deployment pipeline for production. We're building continuous delivery pipeline.

Continuous delivery is a process that ensures our application is production ready. When we are doing a scheduled deployment we can do so with confidence since we know the application has been through a pipeline of tests before-hand.

Next, click on the "task" link on the Production stage. We need to modify this task so that it does not deploy our code into the development slot.



Stage name

Production

Parameters | Unlink all

Azure subscription * | Manage

Microsoft Partner Network

Scoped to subscription 'Microsoft Partner Network'

App type

Linux App

App service name *

node-express-demo

Image Source *

Built-in Image

Runtime Stack *

Node.js 8.0

☐ Deploy to slot

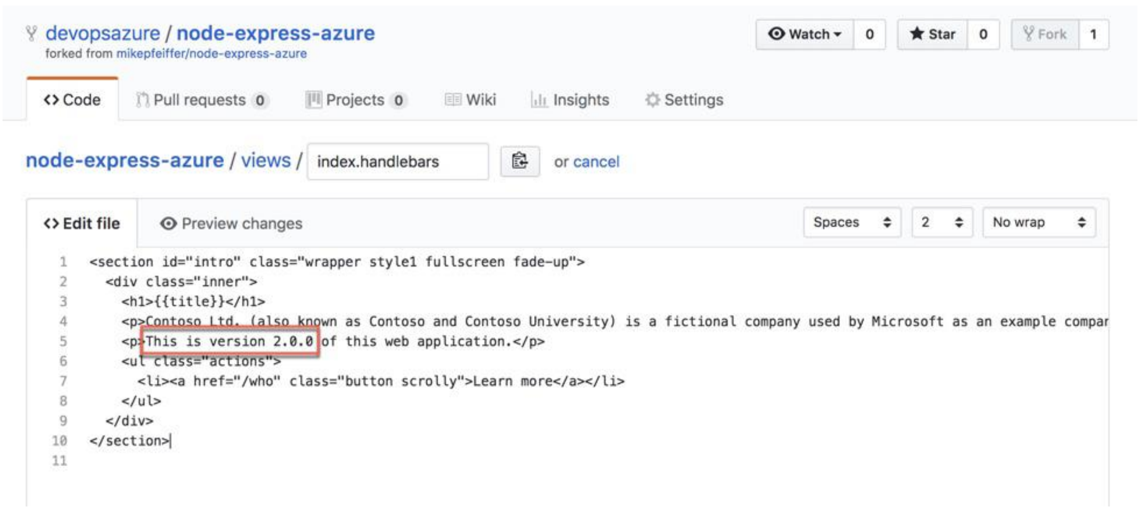
This field is linked to 1 setting in 'Deploy Azure App Service to Slot'

Figure 23. Updating the Deployment Task.

Simply uncheck "slot" and this will infer that the production slot of the web app should be used during the deployment. Click save when complete.

Validating the Pipeline

Navigate to your GitHub account and into the views folder of the demo application. Edit the index.handlebars file to update the app to version 2.0.0.

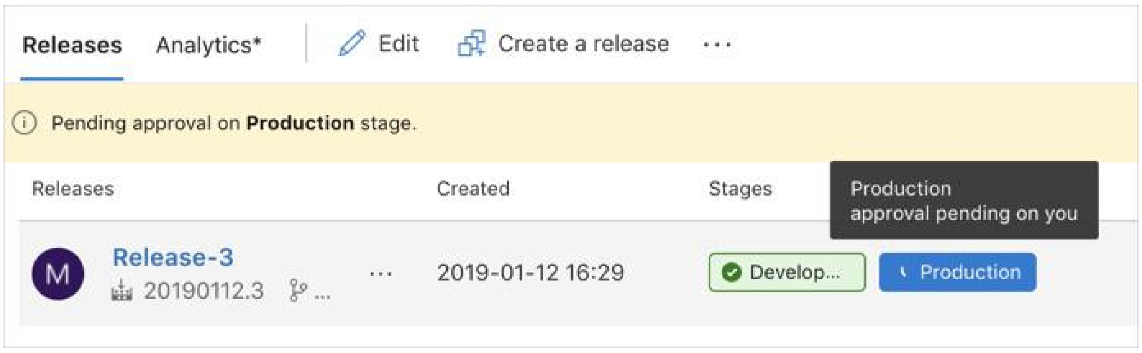

devopsazure / node-express-azure
forked from mikepfeiffer/node-express-azure
Watch 0
Star 0
Fork 1
Code
Pull requests 0
Projects 0
Wiki
Insights
Settings
node-express-azure / views / index.handlebars
or cancel
Edit file
Preview changes
Spaces
2
No wrap

```
<section id="intro" class="wrapper style1 fullscreen fade-up">
  <div class="inner">
    <h1>{{title}}</h1>
    <p>Contoso Ltd. (also known as Contoso and Contoso University) is a fictional company used by Microsoft as an example compar
    <p>This is version 2.0.0 of this web application.</p>
    <ul class="actions">
      <li><a href="/who" class="button scrolly">Learn more</a></li>
    </ul>
  </div>
</section>
```

Figure 24. Radically Modifying the Code Base for the App.

Committing the change in this repo should automatically trigger a build, perform our tests, and publish a deployment package. We can confirm this by reviewing the build status.

After the build, you should see a new release. The development stage should be green indicating that the deployment succeeded. The production stage should be blue and show that it's pending approval.



Pending approval on **Production** stage.

Figure 25. Reviewing the Release Status.

Click approve to kick-off the production deployment.

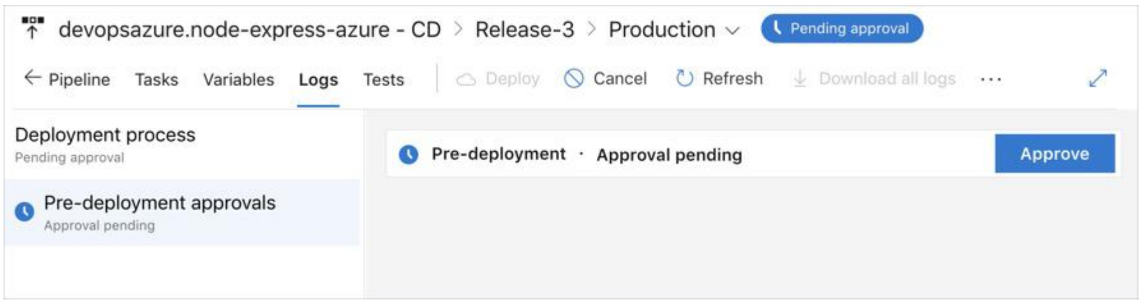


Figure 26. Deploying to Production. Never do this on a Friday!

Go back to your pipeline view and you should see the deployment to production succeeded.

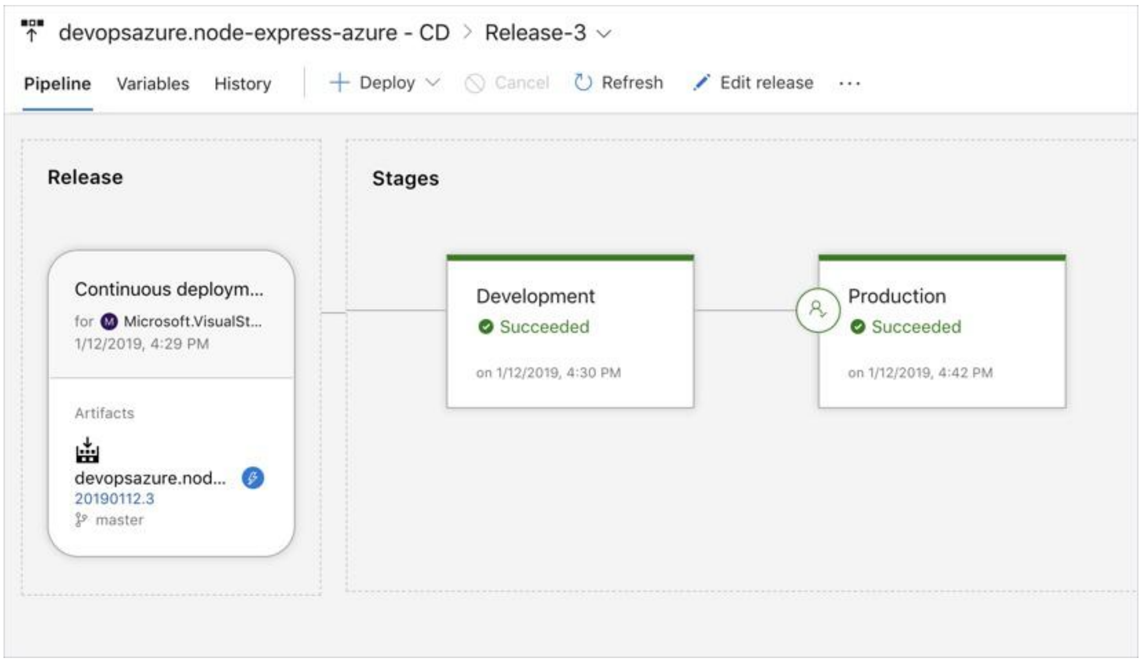
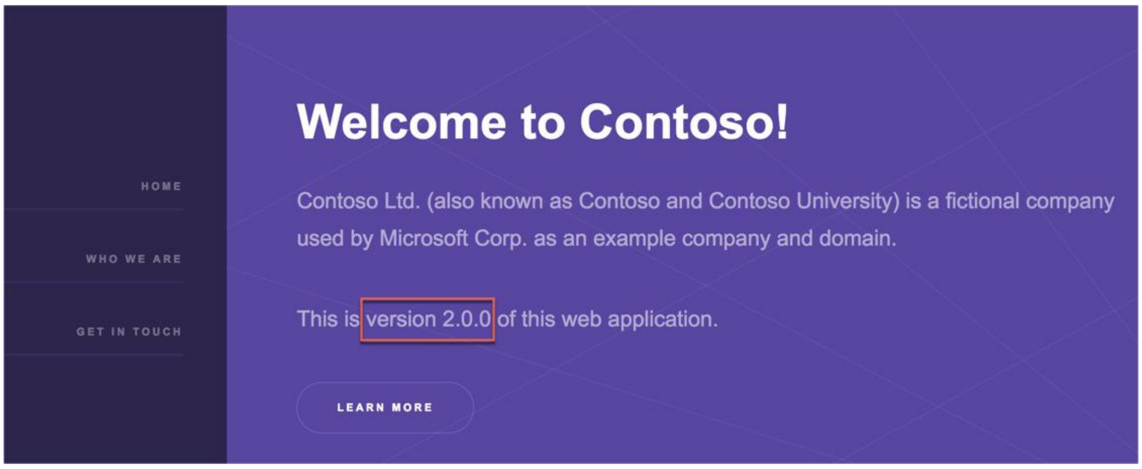


Figure 27. Validating the Status.

Finally, head over to the web app URL for the production slot to confirm the correct version is running.



Contoso Ltd. (also known as Contoso and Contoso University) is a fictional company used by Microsoft Corp. as an example company and domain.

This is version 2.0.0 of this web application.

Figure 28. Reviewing the Version.

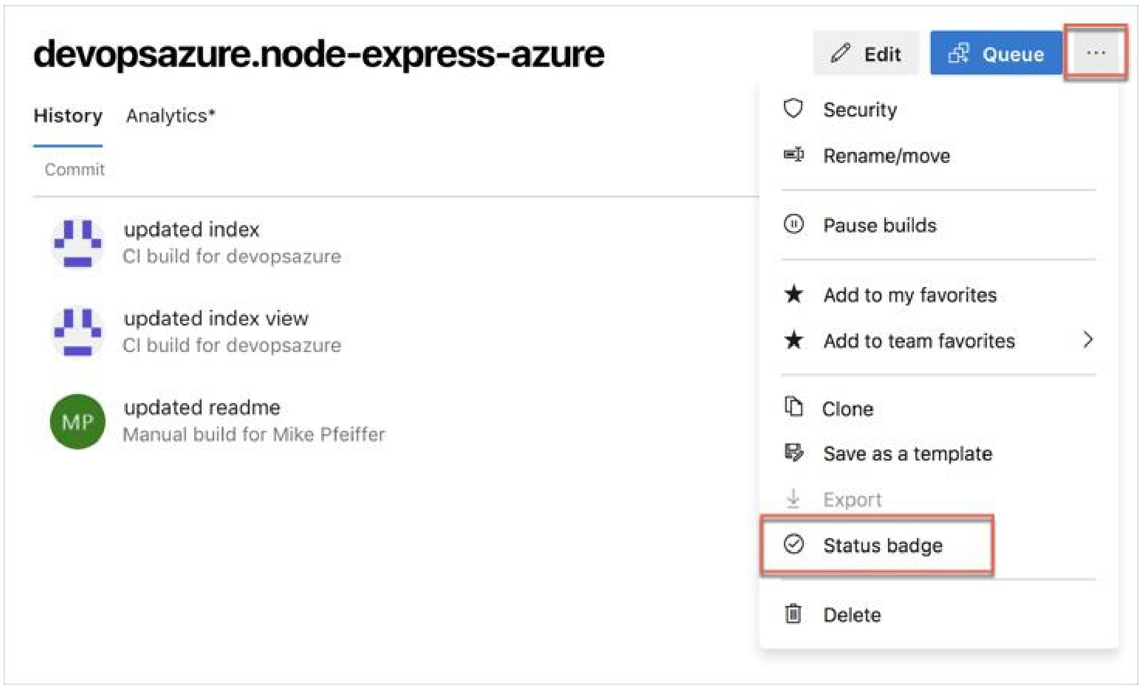
You should see version 2.0.0 on the homepage.

Set up a Build Badge for Your Project

Have you ever seen those build pass/fail badges when browsing projects on GitHub? They're really cool because you can tell at a glance if the code is still working or if it's old and busted.

Let's setup a badge for this project.

Go back to your Builds section and click the status badge button.


devopsazure.node-express-azure

History Analytics*

Commit

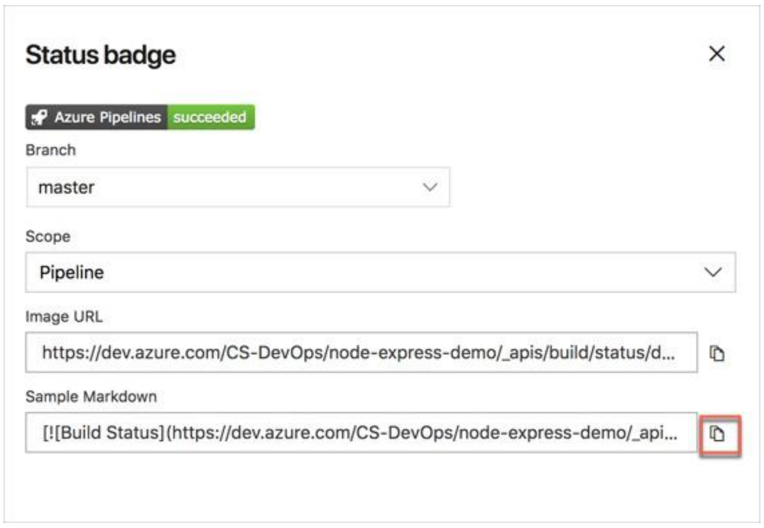
updated index
CI build for devopsazure

updated index view
CI build for devopsazure

updated readme
Manual build for Mike Pfeiffer

Figure 29. Locating the Status Badge Button.

Copy the markdown code for the status badge.


Status badge

Azure Pipelines succeeded

Branch: master

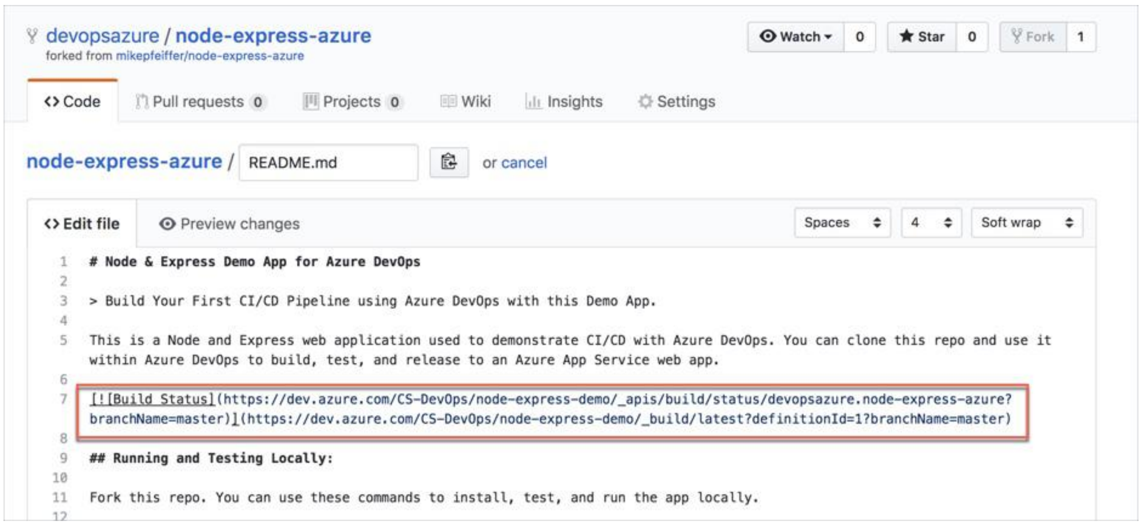
Scope: Pipeline

Image URL: https://dev.azure.com/CS-DevOps/node-express-demo/_apis/build/status/d...

Sample Markdown: [![Build Status](https://dev.azure.com/CS-DevOps/node-express-demo/_api...

Figure 30. Copying the Markdown.

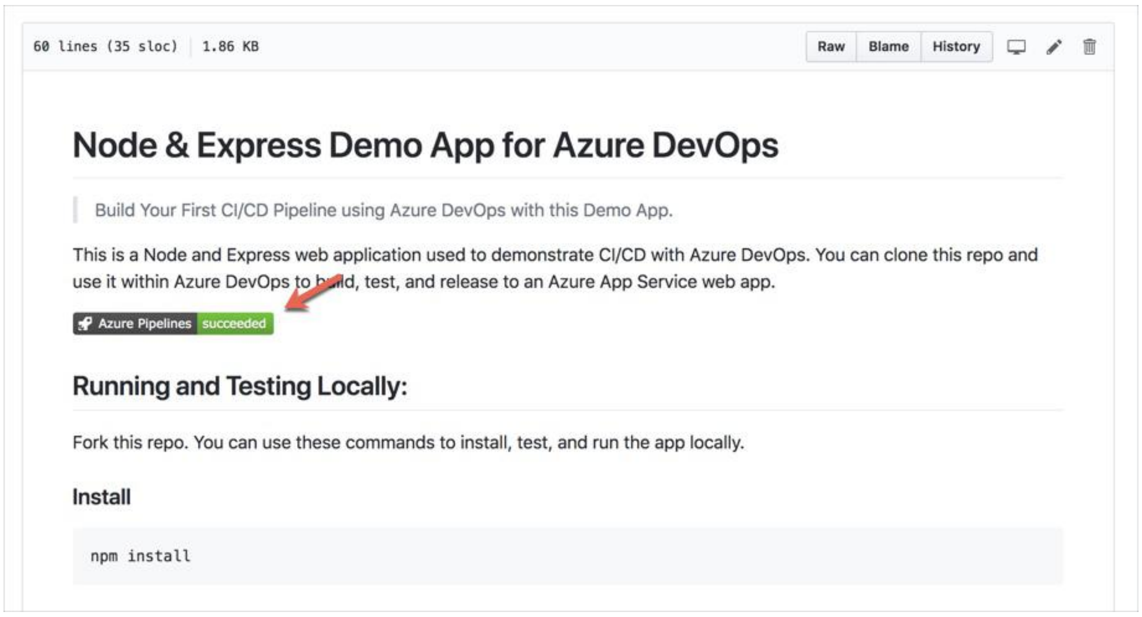
Now, go back to GitHub and modify the README.md file in your node-express-azure repo. Paste the markdown you copied from the status badge page.



```
# Node & Express Demo App for Azure DevOps

> Build Your First CI/CD Pipeline using Azure DevOps with this Demo App.

This is a Node and Express web application used to demonstrate CI/CD with Azure DevOps. You can clone this repo and use it
within Azure DevOps to build, test, and release to an Azure App Service web app.

[![Build Status](https://dev.azure.com/CS-DevOps/node-express-demo/_apis/build/status/devopsazure.node-express-azure?
branchName=master)](https://dev.azure.com/CS-DevOps/node-express-demo/_build/latest?definitionId=1?branchName=master)

## Running and Testing Locally:

Fork this repo. You can use these commands to install, test, and run the app locally.
```

Figure 31. Updating README.md.

Commit the change and view the README. You should see a build passing status icon.



Node & Express Demo App for Azure DevOps

> Build Your First CI/CD Pipeline using Azure DevOps with this Demo App.

This is a Node and Express web application used to demonstrate CI/CD with Azure DevOps. You can clone this repo and use it within Azure DevOps to build, test, and release to an Azure App Service web app.

Running and Testing Locally:

Fork this repo. You can use these commands to install, test, and run the app locally.

Install

```
npm install
```

Figure 32. Badge Status Icon in Action.

Create a Release Stage for Production

Head back over to the Azure DevOps portal and go to Pipelines > Releases. Click on the "Edit" button to modify the pipeline. Highlight the Development stage and click the dropdown to clone the stage.