

You have 1 free story left this month. [Upgrade for unlimited access.](#)

Making MEAN stack work with Azure DevOps



Eason

Sep 20, 2018 · 9 min read ★

MEAN stack (MongoDB + ExpressJS, AngularJS + NodeJS) is most common open-source Javascript software stack for building dynamic web sites and apps. Recently Microsoft announce Azure DevOps to help developer streamline their development life-cycle and help to ship apps faster. This is evolution of long time history of VSTS (Visual Studio Team Services) which make it more open-source software stack development friendly.

So, let me show you how to make MEAN stack working with Azure DevOps to streamline development life-cycle.

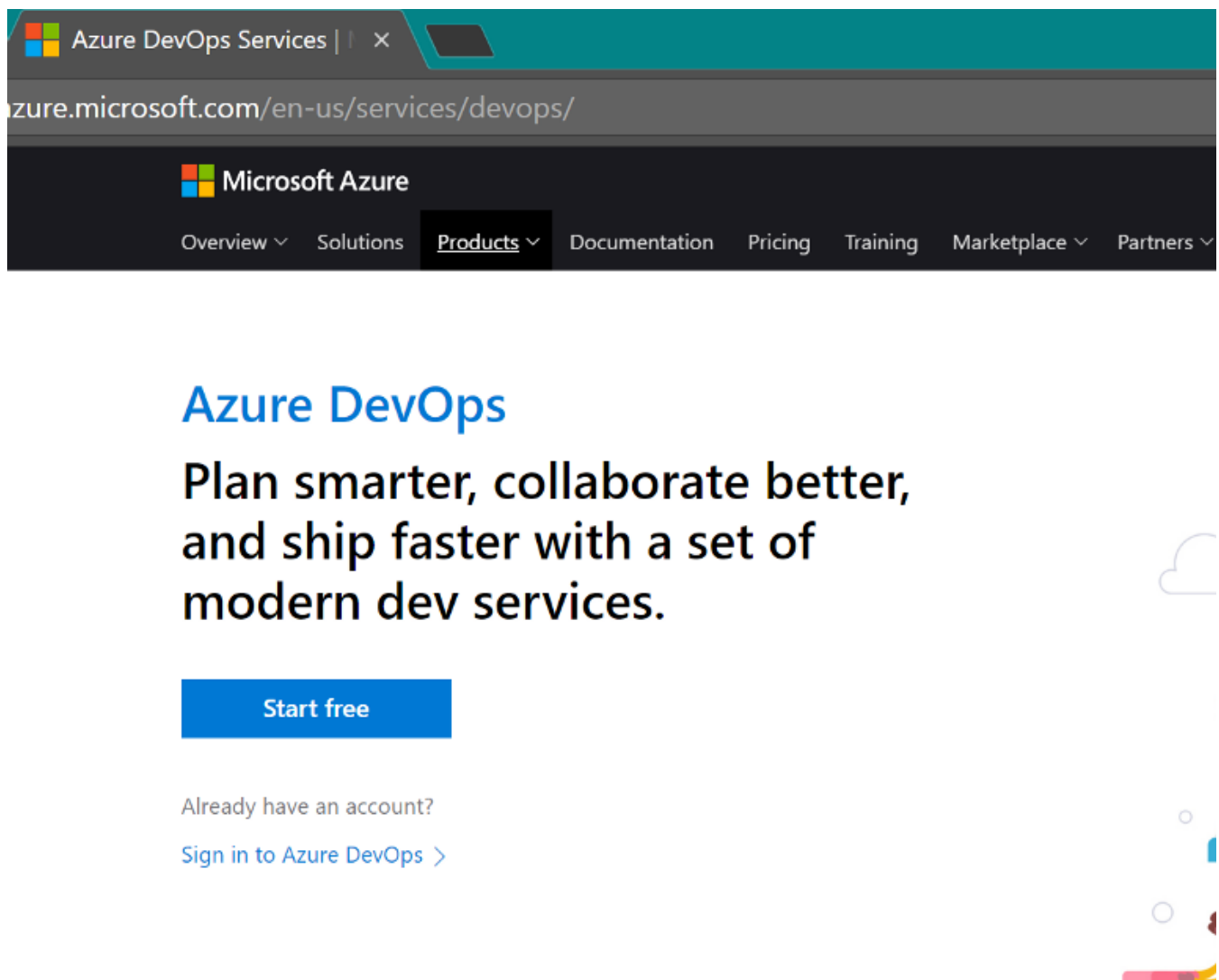
First thing first, is it costly?

Yay! It's free to Open Source Projects, i.e. if you're creating Public Repos, it's allow you to have 10 parallel jobs with unlimited minutes for CI/CD pipeline.

Open source projects	Small Teams	Teams of any size
Free	Free	\$30/mo
Unlimited users and build time	Start free with up to 5 users	10 Users ▼
<ul style="list-style-type: none">• Azure Pipelines: 10 parallel jobs with unlimited minutes for CI/CD• Azure Boards: Work item tracking and Kanban boards• Azure Repos: Unlimited public Git repos	<ul style="list-style-type: none">• Azure Pipelines: 1 hosted job with 1,800 minutes per month for CI/CD and 1 self-hosted job• Azure Boards: Work item tracking and Kanban boards• Azure Repos: Unlimited private Git repos• Azure Artifacts: Package management• Load testing (20,000 VUMs/month)• Unlimited stakeholders	<ul style="list-style-type: none">• Azure Pipelines: 1 hosted job with 1,800 minutes per month for CI/CD and 1 self-hosted job• Azure Boards: Work item tracking and Kanban boards• Azure Repos: Unlimited private Git repos• Azure Artifacts: Package management• Load testing (20,000 VUMs/month)• Unlimited stakeholders• Visual Studio subscribers included free
Start for free	Start for free	Start

How about Private Repos? You can start from small from Free tier. It's give you 1 parallel job with 1,800 minutes per month for CI/CD pipeline. This should be good enough to serve for 1 - 2 small or medium scale open-source projects CI/CD pipeline in most case.

Let's get start the journey... First, let's go to Azure DevOps landing page and click Sign In. Or if you never touch it before, just click Start free and follow few simple steps to make your account login.



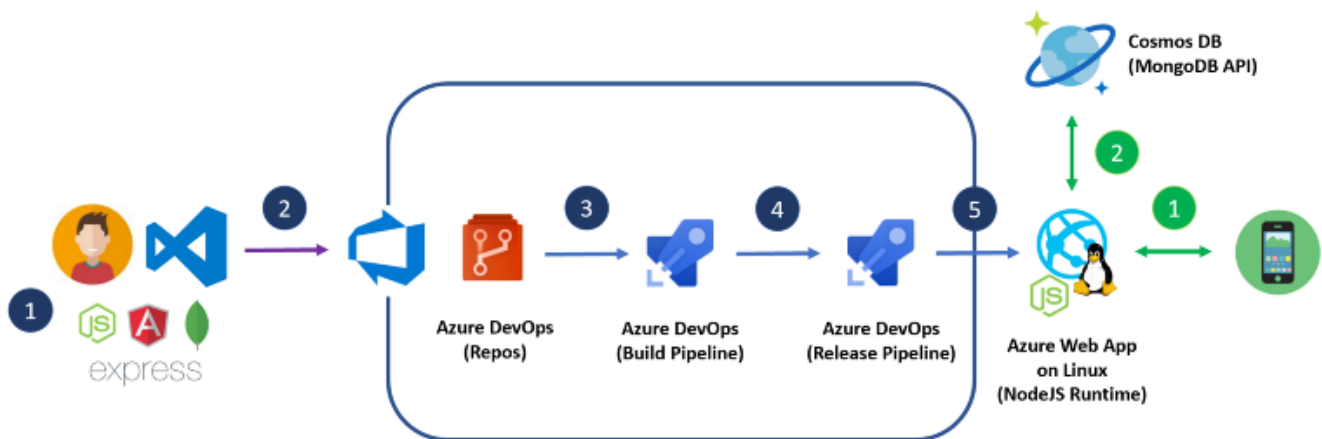
After login successfully, you should able to see Azure DevOps organizations dashboard.



Wait... let me tell you the high-level flows and architecture before going to details... I'll use a simple to-do app which built with MongoDB, ExpressJS, AngularJS and NodeJS for this sharing.

NodeJS & ExpressJS provides the RESTful API. AngularJS provides the frontend and accesses the API. Cosmos DB (MongoDB API) stores like a hoarder.

You can clone my repos from GitHub.



Left to Right...

1. Developer local development environment (Mac / Windows) which installed NodeJS, MongoDB and then npm install required package like Express, Mongoose, etc... And it's also installed free code editor Visual Studio Code as it have lots of open-sources friendly extension debugger.
2. Git Stage, Commit and Push (by command or GUI via VS Code) from local to Azure DevOps Repos.
3. Build Pipeline is being triggered as there are new commit from Repos.
4. Deploy Pipeline is being triggered as there are new Build from Build Pipeline.
5. Deploy Pipeline deploy the Build Artifact to Azure Web App on Linux which has pre-installed NodeJS runtime.

Right to Left

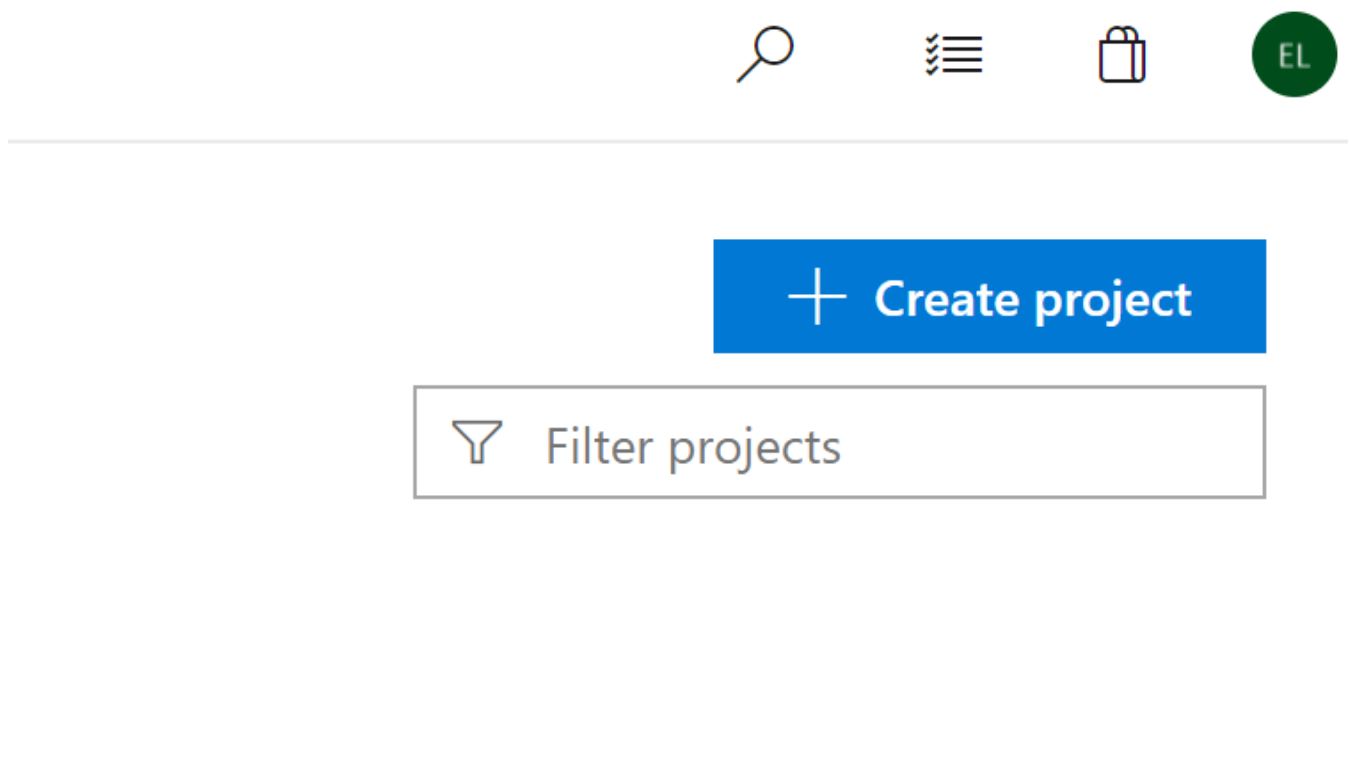
1. Access web app from mobile device.
2. Web App connect to No-SQL Database (Cosmos DB) via MongoDB API and show the results.

Little bit long sharing, we have 4 stages to make it works...

All right, let's get started...

Stage 1: Push your local code to Azure DevOps Repos

1. Back to Azure DevOps organizations dashboard and click "Create Project".



2. Type your project name and select Visibility you want. For this sharing, I'll use Private one.

Create new project



Project name *

mean-devops-sample02



Description

Visibility



Public

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

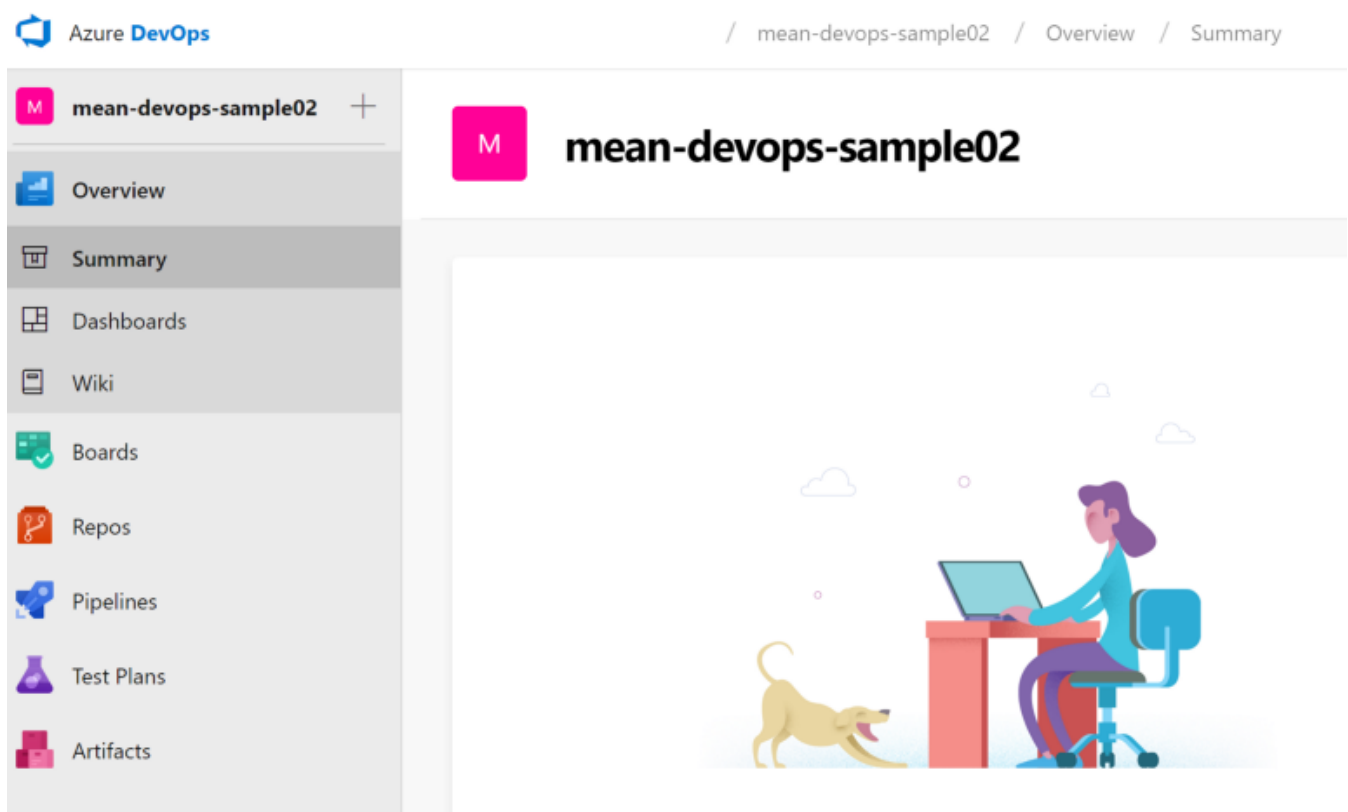


Private

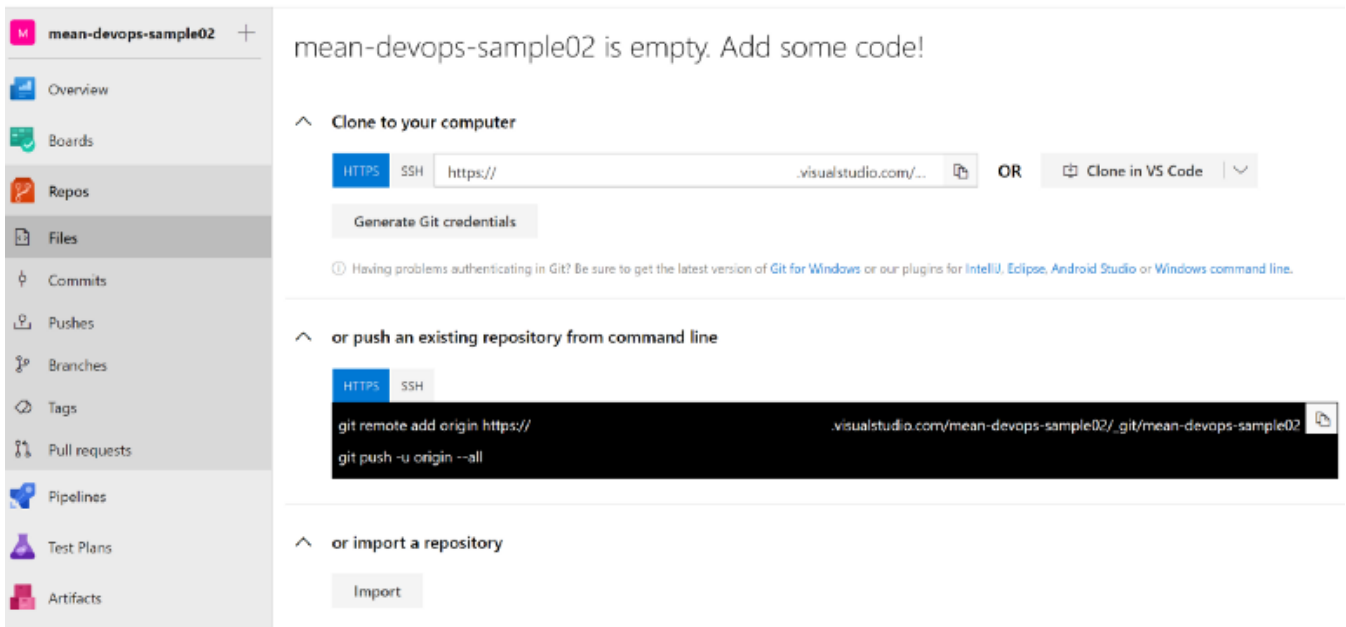
Only people you give access to will be able to view this project.

▼ Advanced

3. Wait for few seconds, you'll going to your project Dashboard. Then click "Repos".

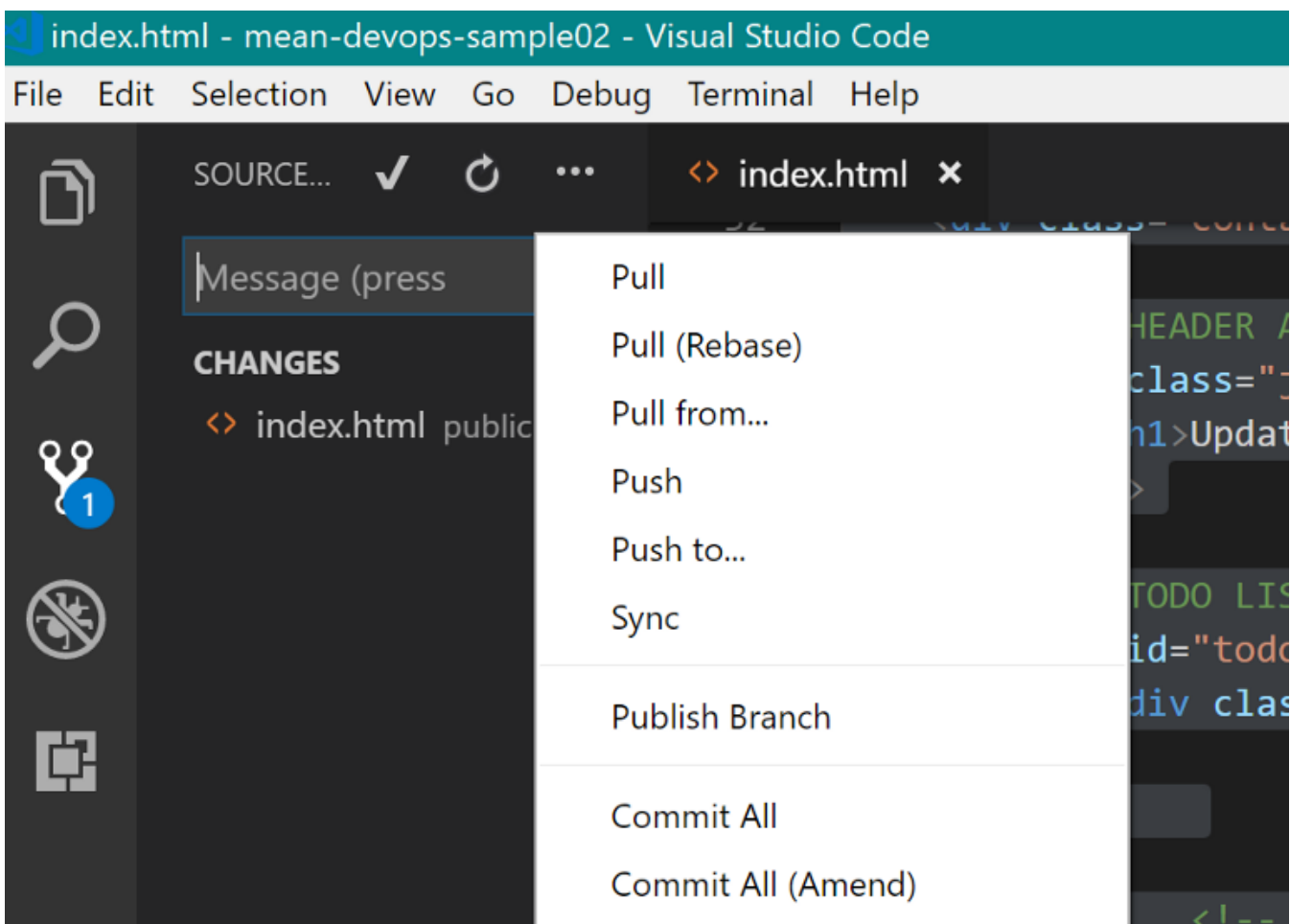


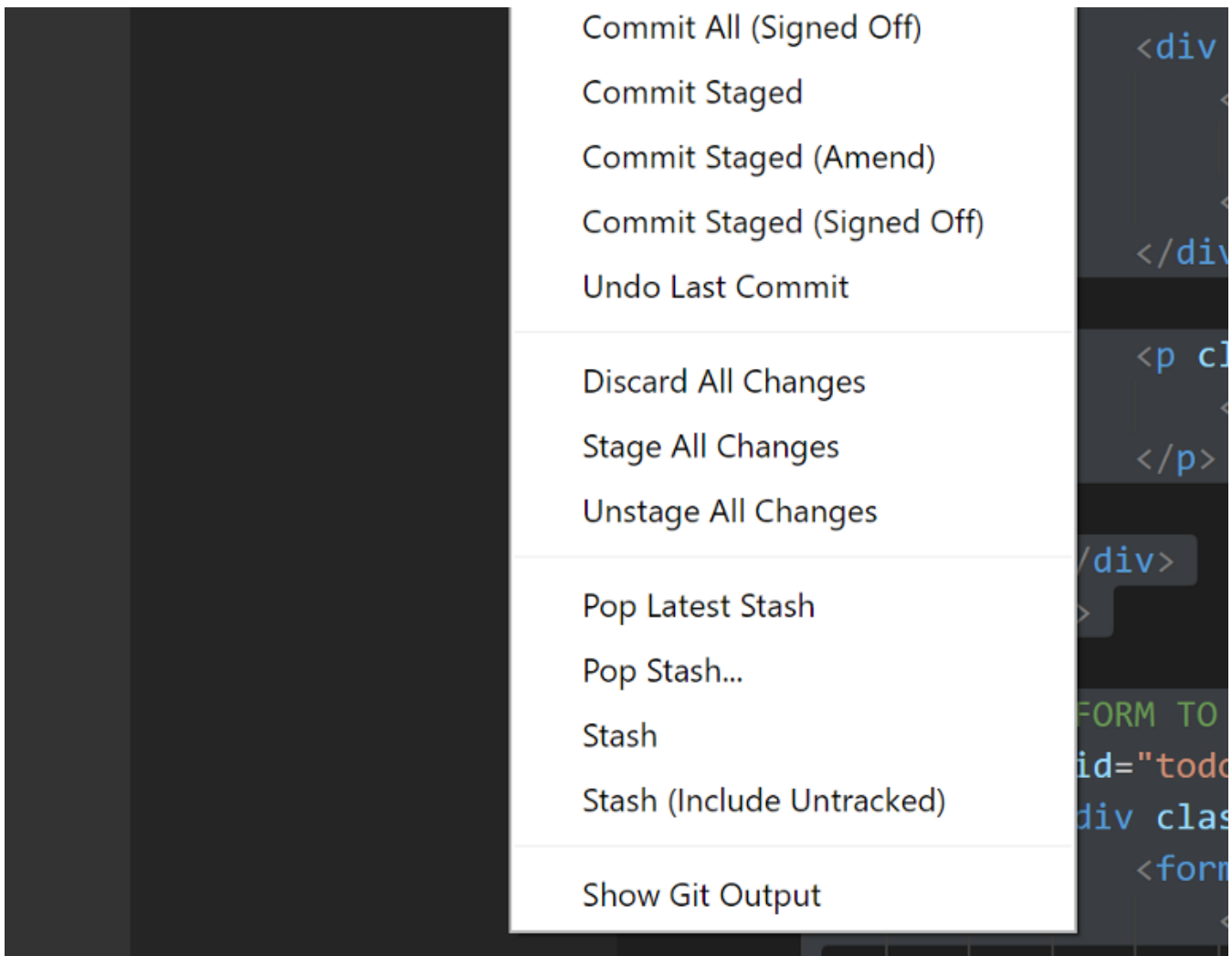
4. Obviously, it's a blank repos, you can push your existing repos via git remote add and push. Or if you're starting a new project, click "Clone in VS Code" should be a good idea, it will automatically make your VS Code to deal with this new repos. This time, I will use this lazy idea :)



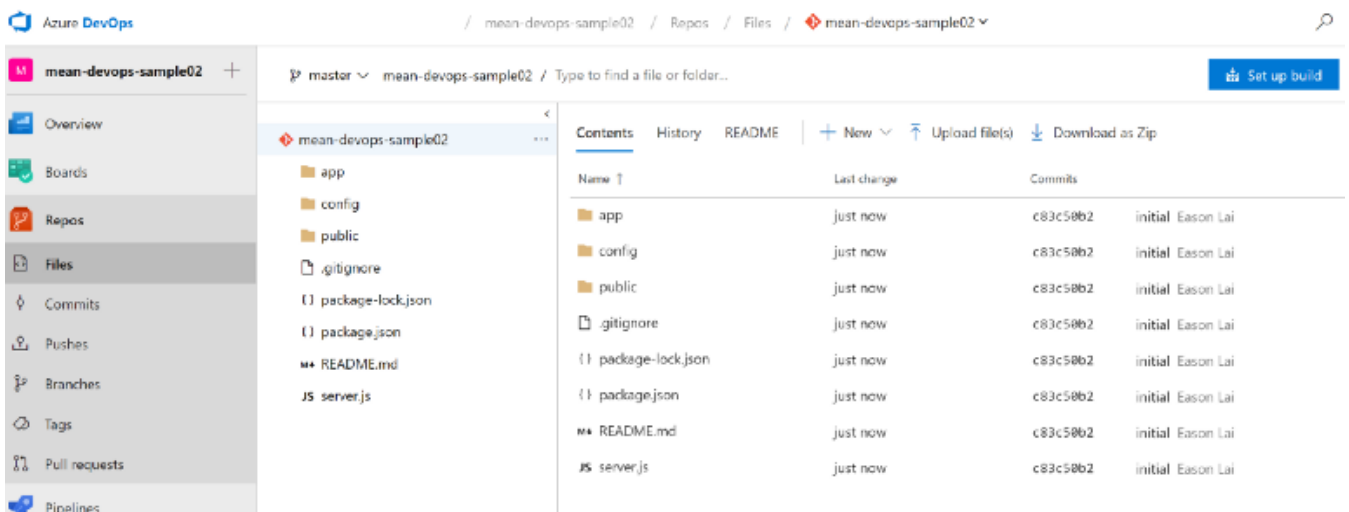
5. Back to VS Code, finish your code, debug, make it works, then click on “Source Control” button. You’re not necessary to type any command for commit your code to repos, just click Stage, Commit (with your message), Push...

- For Node project, you should make a .gitignore file to ignore pushing “node_modules” folder. When we setup a Build Pipeline in Azure DevOps, we’ll make a “npm install” task to install required packages.





6. Wait for few seconds, back to your Repos in Azure DevOps. All you code should be appeared.



Stage 2: Setup Build Pipeline which triggered from Repos

1. Actually, there are “Set up build” button in your Repos page, just click on it.

sample02 / Type to find a file or folder...

Set up build

Contents	History	README	+ New	Upload file(s)	Download as Zip
Name ↑	Last change	Commits			
app	just now	c83c50b2	initial	Eason Lai	
config	just now	c83c50b2	initial	Eason Lai	

2. There are lots of template build task packages. This time, I will use “Node.js With gulp” template, click on Apply.

3. There are 4 jobs will automatically add for this Pipeline. First, make a meaningful name of your Build Pipeline, I use “MEANBuild” this time. Then select “Agent pool”, I use “Hosed VS2017” this time which is Windows OS based build server.

4. Next, click “Get sources”, actually, you’re not necessary to configure anythings as you click this Build Pipeline setup from Repos directly. Definitely, if you can further set it up, e.g. branch.

The screenshot shows the Azure DevOps Build Pipeline configuration interface. The pipeline is named "MEANBuild". The "Get sources" task is selected, and its configuration is shown on the right. The configuration includes:

- Select a source:** Azure Repos Git (selected), GitHub, GitHub Enterprise, Bitbucket Cloud, and External Git.
- Team project:** mean-devops-sample02
- Repository:** mean-devops-sample02
- Default branch for manual and scheduled builds:** master
- Clean:** false

The task list on the left includes:

- Get sources (selected)
- Agent job 1
- npm install
- Run gulp task
- Archive files
- Publish artifacts: drop

5. Click “npm install” task, by default, if your “package.json” is located in your repos root, you’re not necessary specify. Or you can click ... to specify if you’re putting in specific directory.

The screenshot shows the Azure DevOps Build Pipeline configuration interface. The pipeline is named "MEANBuild". The "npm install" task is selected, and its configuration is shown on the right. The configuration includes:

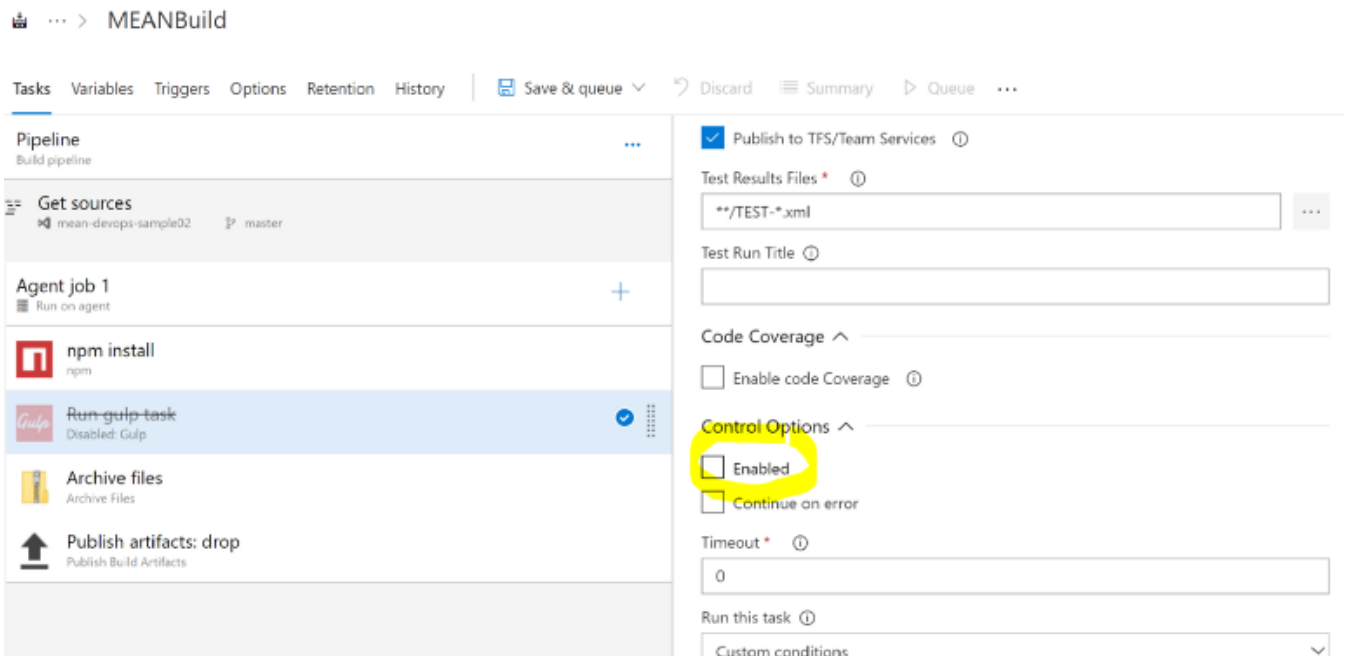
- Version:** 1.*
- Display name:** npm install
- Command:** install
- Working folder with package.json:** (empty)
- Custom registries and authentication:** (empty)
- Advanced:** (empty)

The task list on the left includes:

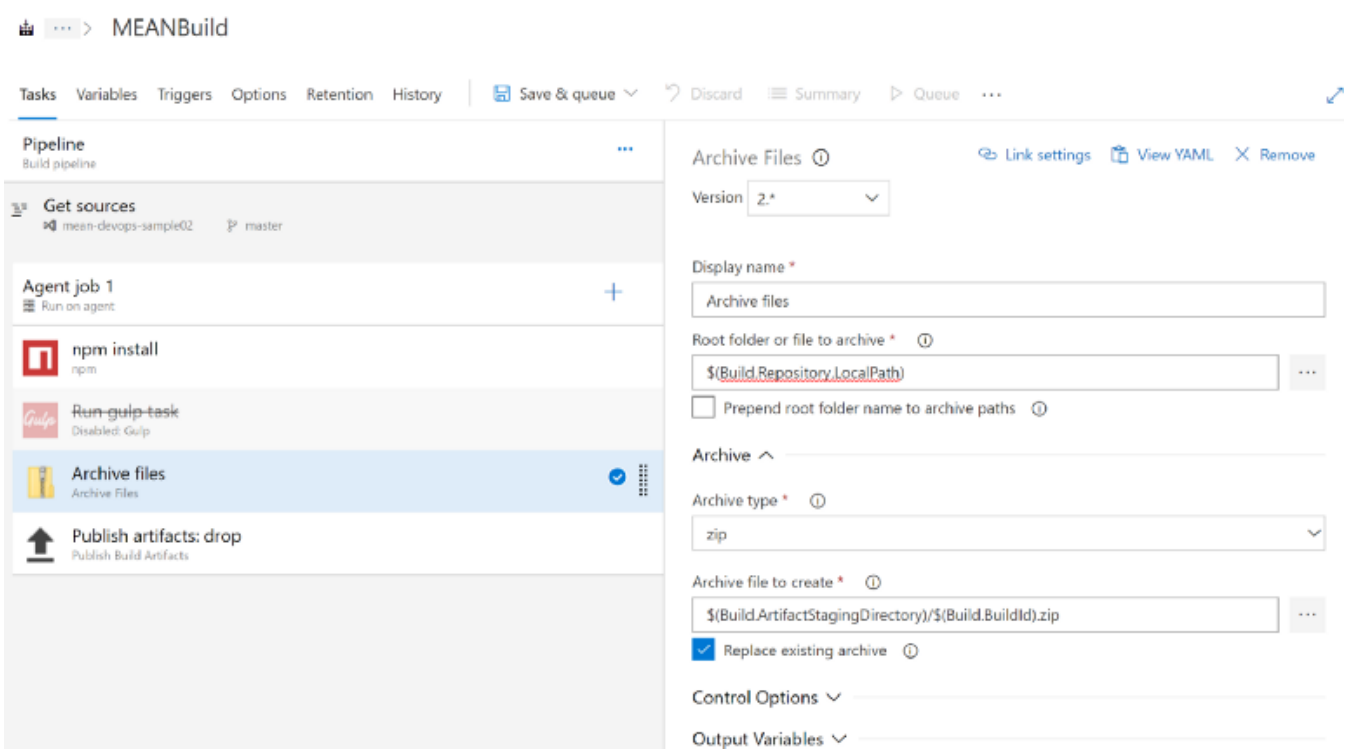
- Get sources
- Agent job 1
- npm install (selected)
- Run gulp task
- Archive files
- Publish artifacts: drop



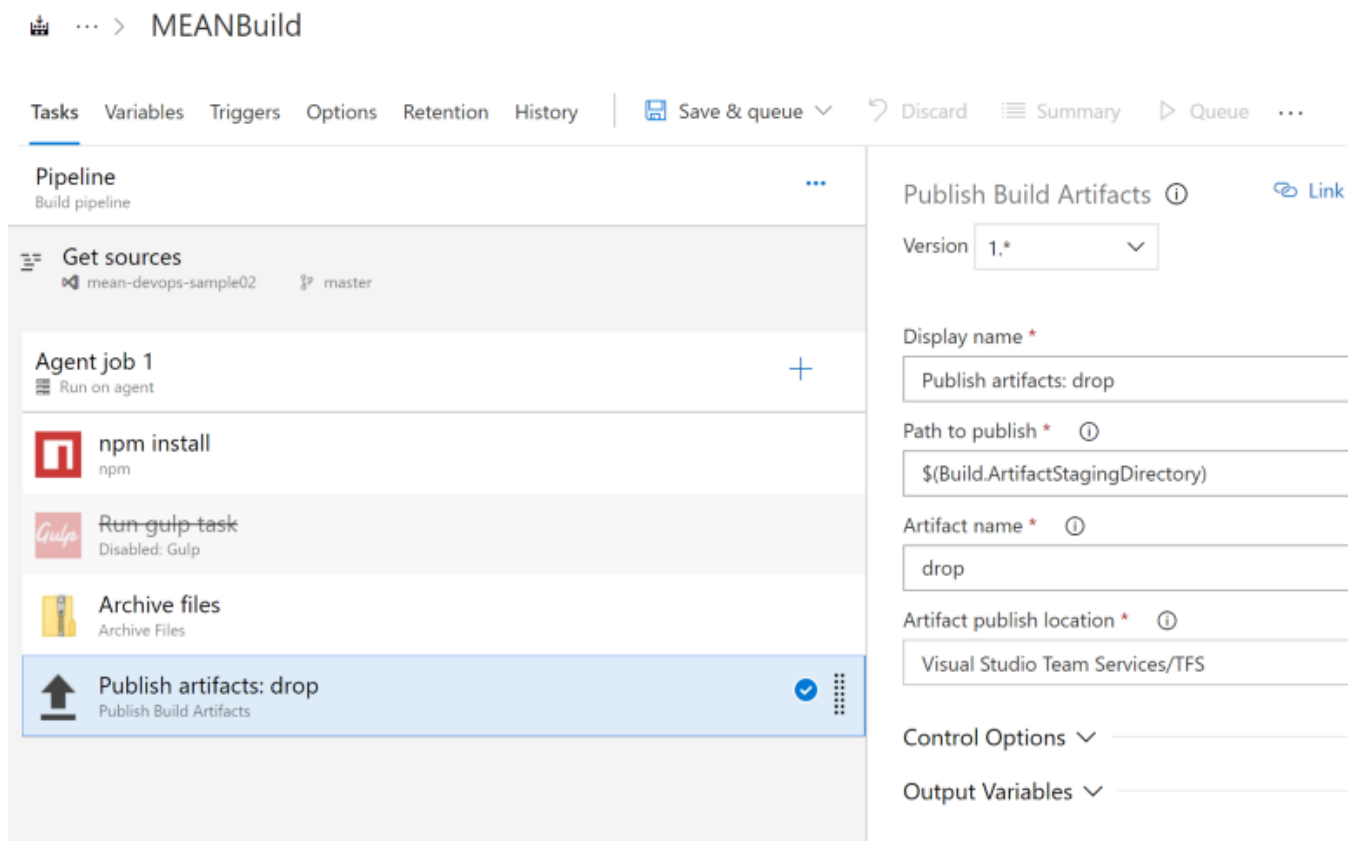
6. Click “Run gulp” task, as I’ll not use guip this time, you can click “Control Options” and uncheck the “Enabled” box. Then this task will be skip to run.



7. Click “Archive file” task, change the “Root folder or file to archive” to “\$(Build.Repository.LocalPath)”. This task will zip the entire repos folder after necessary packages being installed.



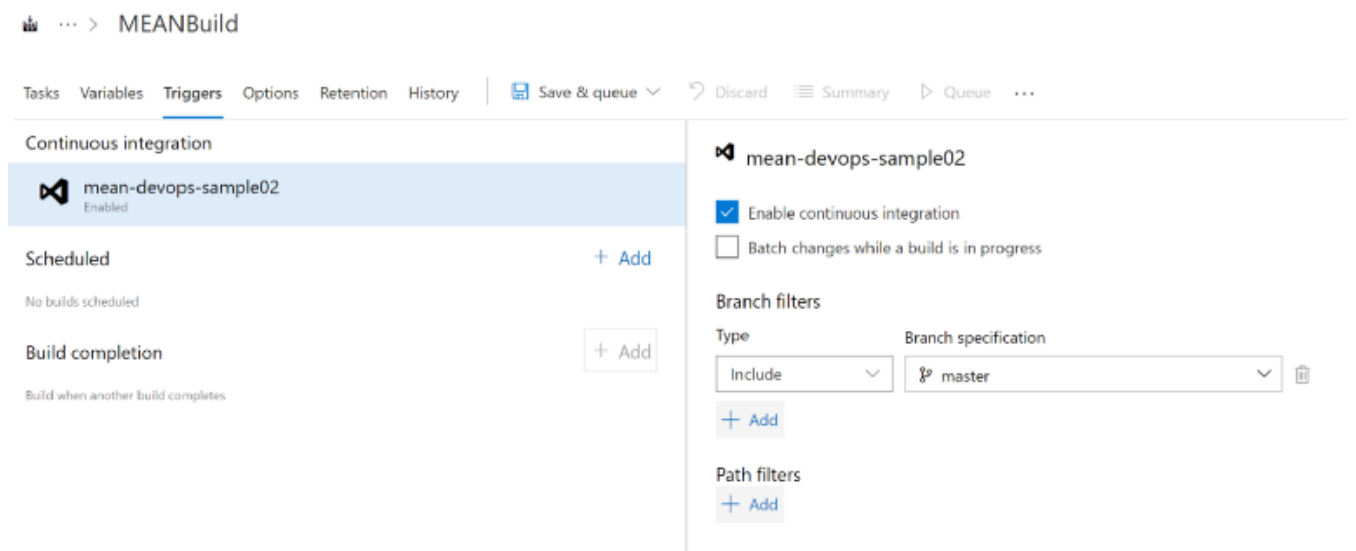
8. Click “Publish artifacts” task, this task will publish your previous task archive zip file to build artifact staging directory’s “drop” folder.



The screenshot shows the Azure DevOps Build Pipeline configuration for a pipeline named "MEANBuild". The "Tasks" tab is active, displaying a list of tasks in the pipeline: "Get sources", "Agent job 1", "npm install", "Run gulp task", "Archive files", and "Publish artifacts: drop". The "Publish artifacts: drop" task is selected and highlighted. To the right of the task list, the configuration for the "Publish Build Artifacts" task is shown. It includes fields for "Version" (set to "1.*"), "Display name" (set to "Publish artifacts: drop"), "Path to publish" (set to "\$(Build.ArtifactStagingDirectory)"), "Artifact name" (set to "drop"), and "Artifact publish location" (set to "Visual Studio Team Services/TFS"). There are also sections for "Control Options" and "Output Variables".

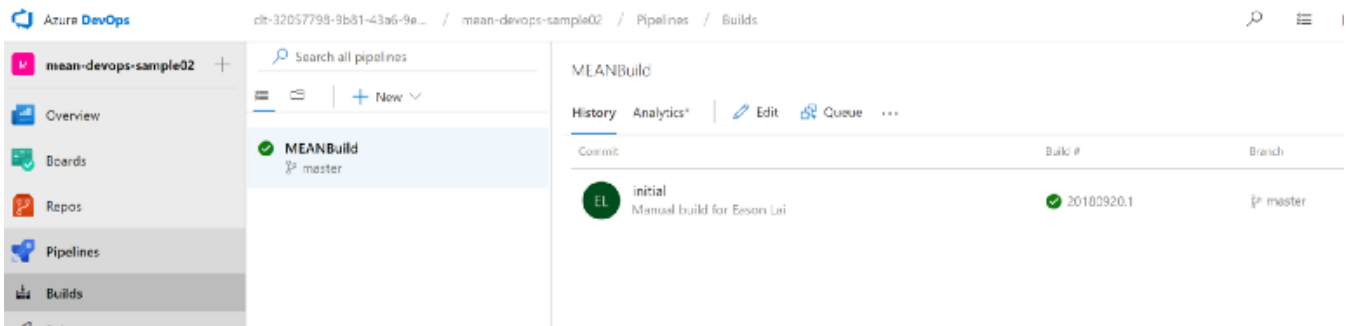
9. Click “Triggers” tab, check the “Enable continuous integration” box. Then if there are any new commit into master branch, it will automatically trigger this Build Pipeline “MEANBuild”.

Click “Save & queue” button, then it will manual trigger this Build Pipeline “MEANBuild” initially.



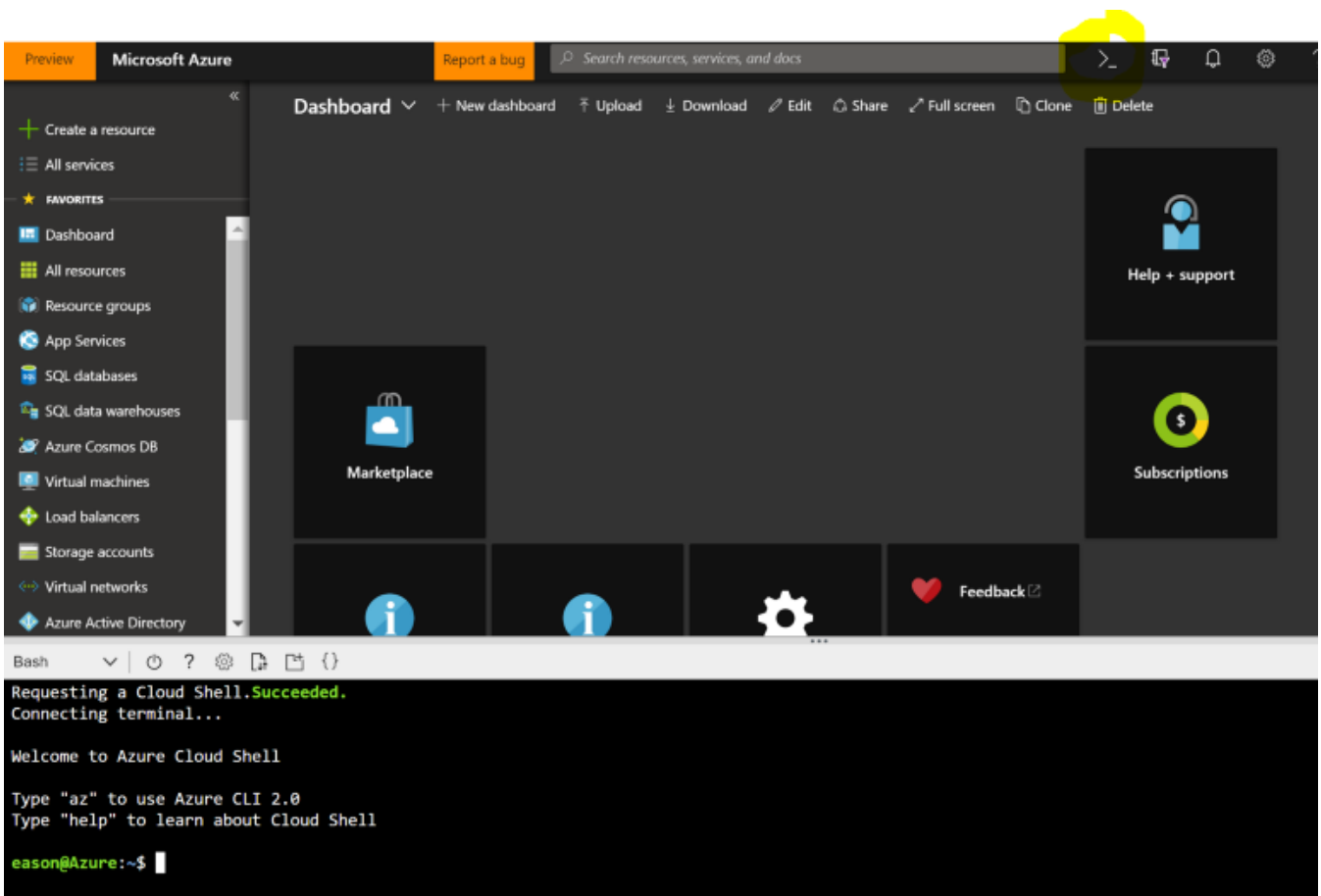
The screenshot shows the Azure DevOps Build Pipeline configuration for a pipeline named "MEANBuild", specifically the "Triggers" tab. The "Continuous integration" section is expanded, showing the "mean-devops-sample02" trigger. The "Enable continuous integration" checkbox is checked. Below it, the "Branch filters" section shows a filter for the "master" branch. The "Path filters" section is also visible.

10. Wait for while, then you should see your Build Pipeline “MEANBuild” successfully build.



Stage 3: Prepare Azure Web App on Linux with NodeJS runtime & Cosmos DB environment

1. Most simple way to create resources in Azure is via Azure Portal or Azure CLI. I'll use both way for this sharing as demonstration.
2. If you like command, but don't want to install Azure CLI on your machine. Azure Portal have Cloud Shell terminal, then you can run the Azure CLI from there directly.



3. Let's create Azure Web App on Linux via Azure CLI first.

Use below command to create new resource group for your workload deployment.

```
az group create --l LOCATION_YOU_LIKE --n GROUP_NAME_AS_YOU_LIKE
```

e.g. `az group create --l southeastasia --n meanappgroup01`

Use below command to create App Service Plan which is running under Linux.

```
az appservice plan create --name PLAN_NAME_AS_YOU_LIKE --resource-group  
GROUP_NAME_YOU_JUST_CREATED --sku B1 --is-linux
```

e.g. `az appservice plan create --name linuxplan01 --resource-group meanappgroup01 -
-sku B1 --is-linux`

Use below command to create Web App under newly created App Service Plan

```
az webapp create --resource-group GROUP_NAME_YOU_JUST_CREATED --plan  
PLAN_NAME_YOU_JUST_CREATED --name WEB_APP_NAME_YOU_LIKE --runtime  
"RUNTIME_YOU_LIKE"
```

e.g. `az webapp create --resource-group meanappgroup01 --plan linuxplan01 --name
meantodosample01 --runtime "node|8.1"`

Yay, at this point you have successfully created Azure App Service on Linux with NodeJS runtime for coming Azure DevOps deployments.

* There are lots of create pre-build runtime, e.g. Java, PHP, Python in various version. You can run the command "`az webapp list-runtimes`" to obtain the full list of latest available runtime.

4. Cosmos DB, I'm not trying to go through few simple steps for Cosmos DB via Azure Portal here. You can refer to this official article here. Remember to select API = MongoDB.

5. After Cosmos DB account creation, going into that resource, click "Connection String" in Settings. Copy the "Primary Connection String" values.



Azure Cosmos DB account

- Connection String

Search (Ctrl+/) <<

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Quick start
- Data Explorer

Settings

- Connection String
- Preview Features
- Replicate data globally
- Default consistency
- Firewall and virtual networks
- Locks
- Automation script

Get started faster

Read-write Keys Read-only Keys

HOST documents.azure.com

PORT 10255

USERNAME

PRIMARY PASSWORD

SECONDARY PASSWORD

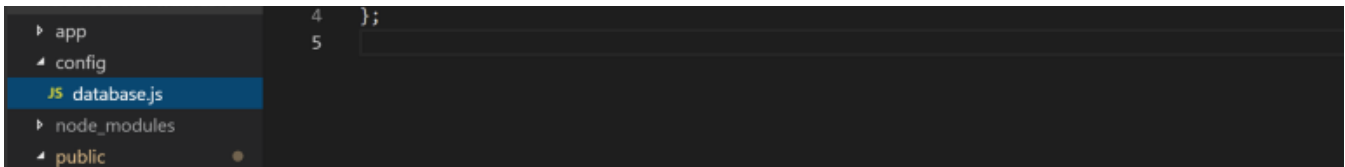
PRIMARY CONNECTION STRING mongodb://q

SECONDARY CONNECTION STRING mongodb://q

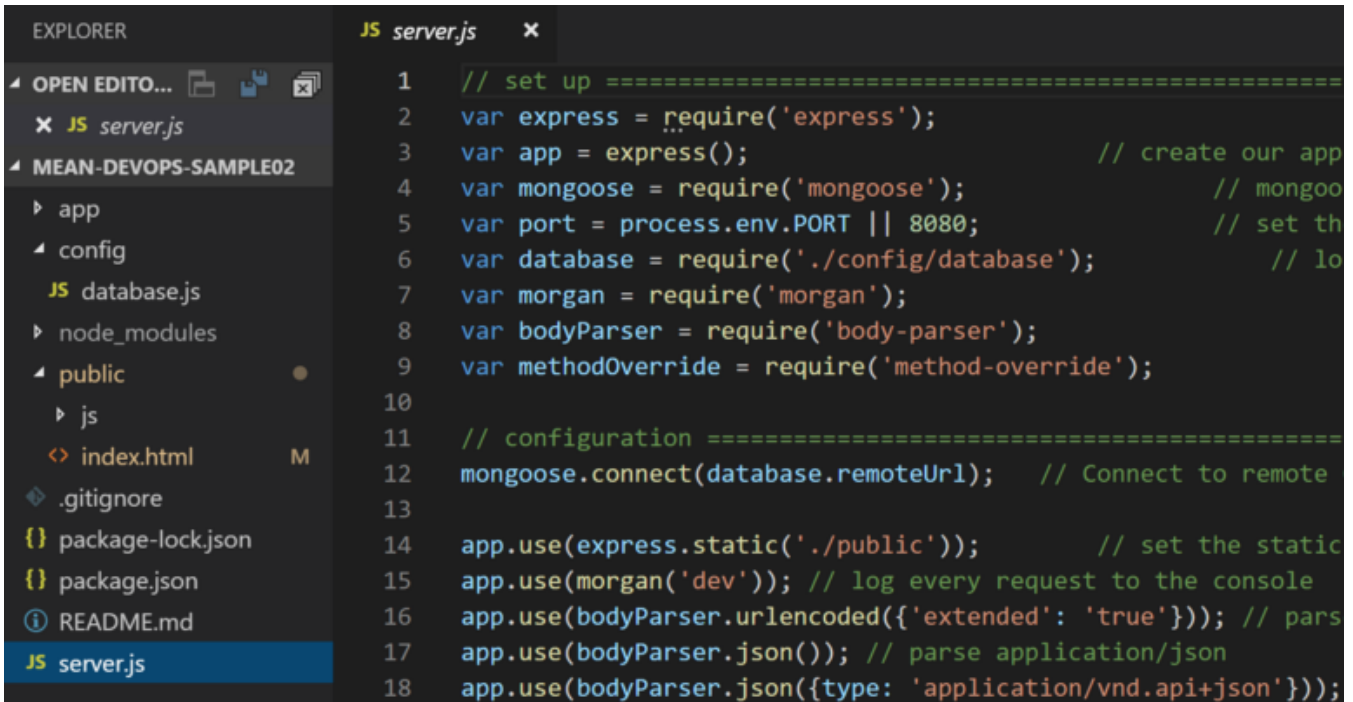
SSL true

6. Back to VS Code, open database.js under config folder. Paste your “Primary Connection String” values into “remoteUrl”.

```
1 module.exports = {
2   remoteUrl: 'mongodb://nodeuser:nodeuser@cosmosdbname.documents.azure.com:10255/',
3   localUrl: 'mongodb://localhost/mean-devops-sample02'
}
```



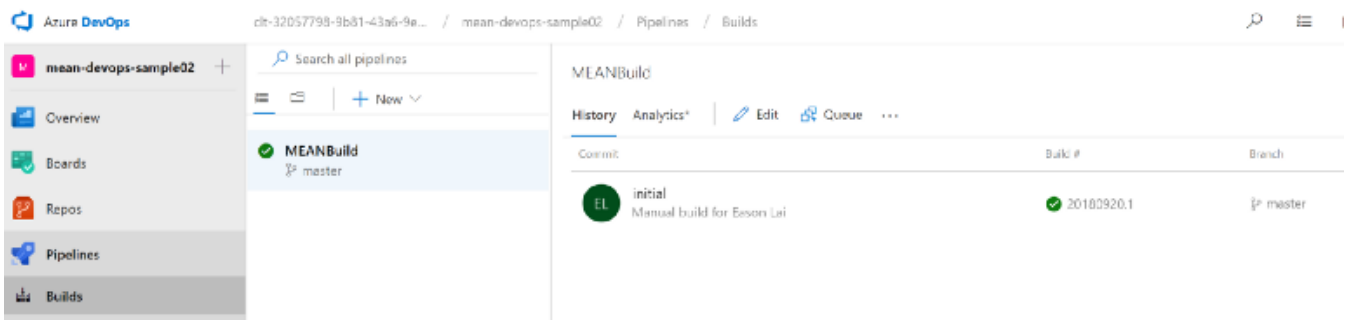
In server.js, make sure line 12' mongoose connection is point to remoteUrl. If point to “localUrl”, it will connect to your local MongoDB environment.



Save, then Stage, Commit (with your message), Push... from Source Control panel. Then, after your code push to Repos in Azure DevOps, it will automatically run your configured Build Pipeline “MEANBuild”. It will give you verification on CI is working as expected.

Stage 4: Setup Release Pipeline which triggered from Build Pipeline

1. Click on your successful build.



2. Click “Release” button.

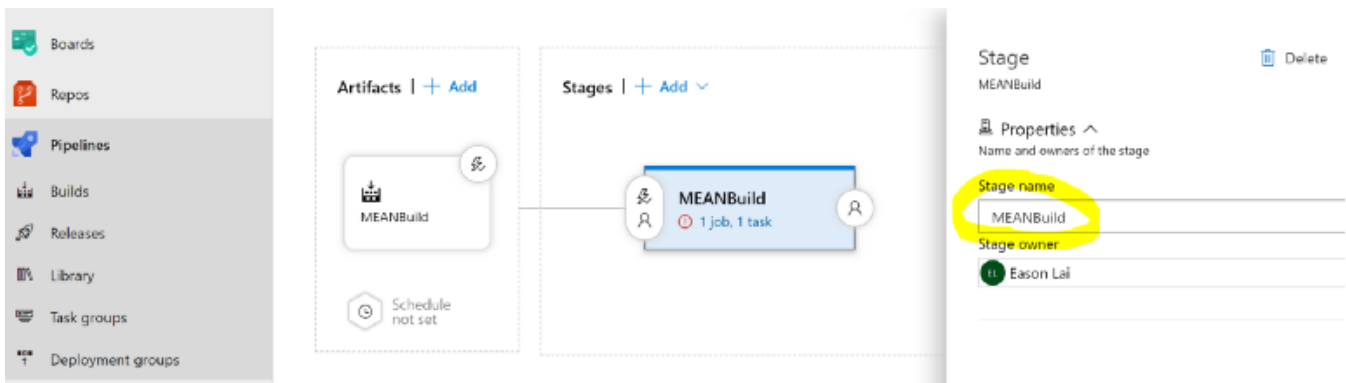
The screenshot shows the Azure DevOps interface for a pipeline named 'mean-devops-sample02'. The left sidebar contains navigation links: Overview, Boards, Repos, Pipelines, Builds, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area displays the 'MEANBuild 20180920.1' pipeline, which is a manual build. The 'Progression' section shows three steps: 1. 'Build artifacts published' (drop, File container), 2. 'Build pipeline succeeded' (0 error(s) / 0 warning(s)), and 3. 'Manually queued' (Eason Lai requested 9 minutes ago). The pipeline is currently in a 'Queue' state.

3. There are lots of template deployment tasks. This time, I will use “Deploy a Node.js app to Azure App Service” template, click on Apply.

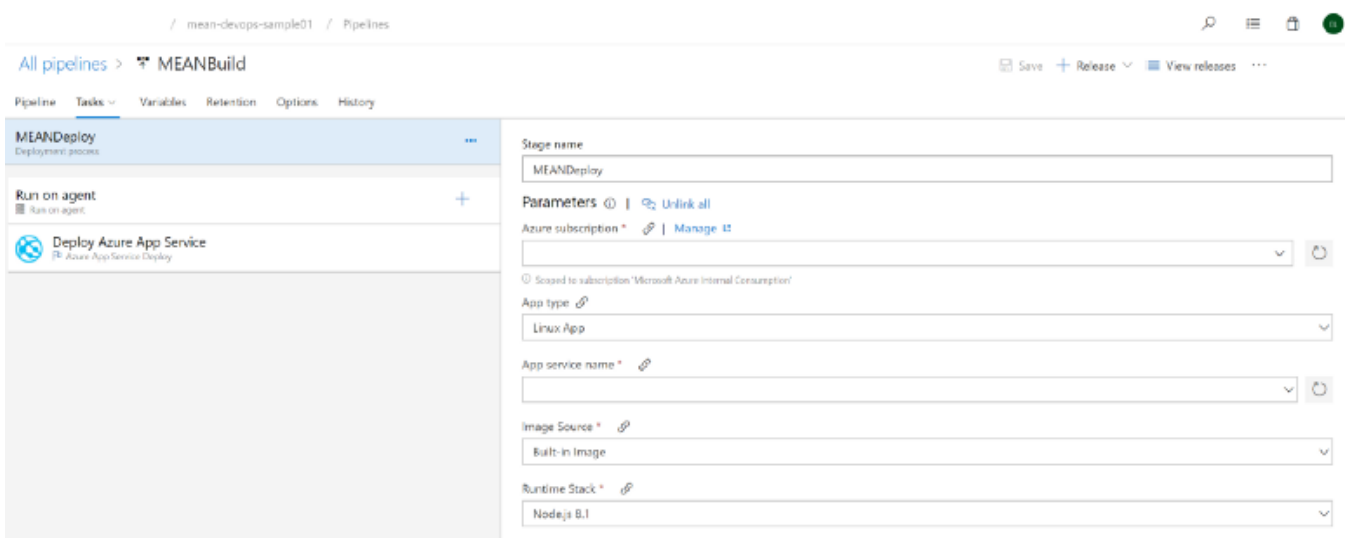
The screenshot shows the 'Select a template' dialog in Azure DevOps. The left sidebar shows the 'MEANBuild - CD' pipeline. The main area displays a list of templates under the 'Featured' section. The templates are: 'Azure App Service deployment', 'Deploy a Java app to Azure App Service', 'Deploy a Node.js app to Azure App Service' (highlighted with an 'Apply' button), 'Deploy a PHP app to Azure App Service and Azure Database for MySQL', and 'Deploy a Python app to Azure App Service and Azure database for MySQL'.

4. Give a name to your Deployment Stage and Release Pipeline name.

The screenshot shows the Azure DevOps interface for the 'mean-devops-sample02' pipeline. The left sidebar shows the 'MEANBuild' pipeline. The main area displays the 'MEANBuild' pipeline name, which is highlighted with a yellow circle. The pipeline is currently in a 'Queue' state.



5. Click on “Tasks” tab. Azure DevOps has deeply integration with Azure. So you can directly select your owned Azure subscription over here. After select right Azure subscription which you want to deploy, click Authorize.



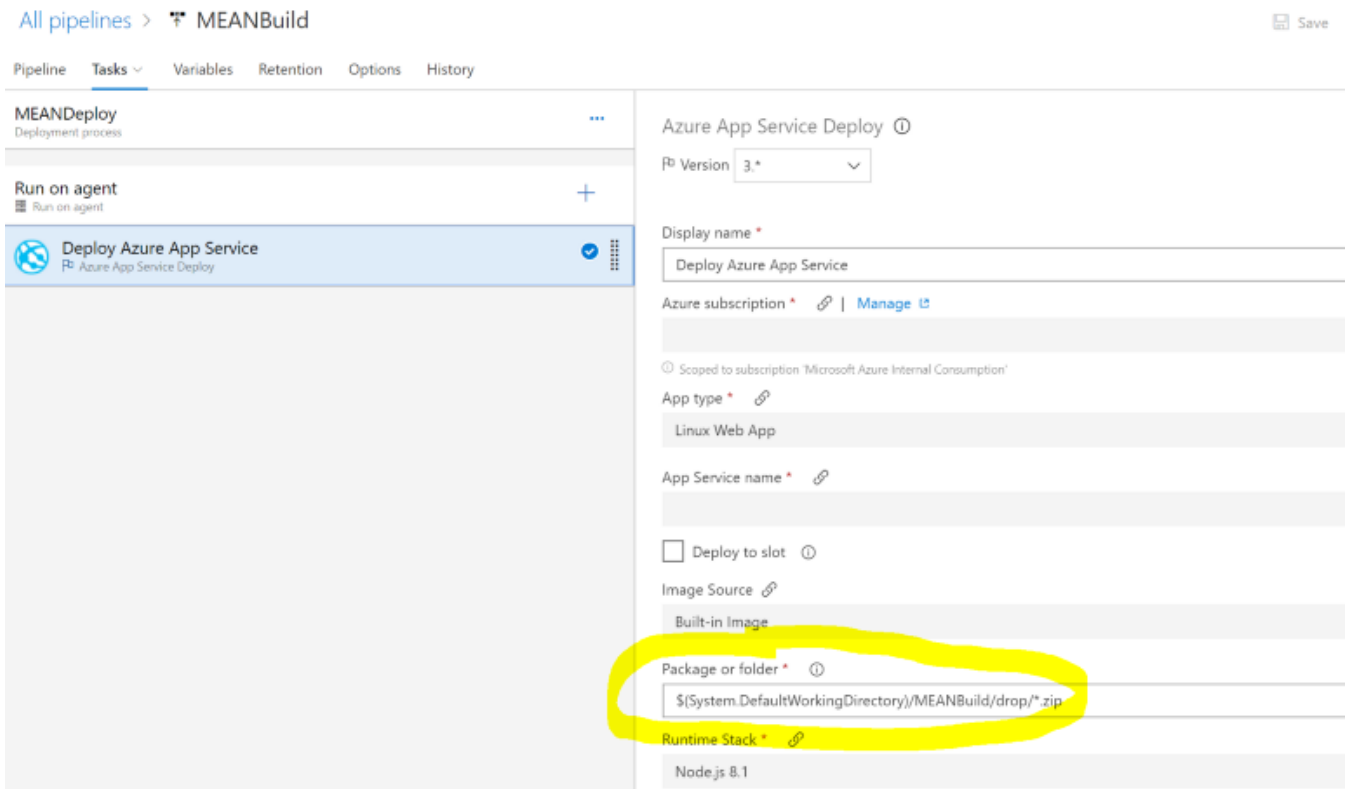
- In App type, select “Linux App” as we target to deploy our MEAN stack app into Azure App Service on Linux.
- In App Service Name, you should able to see your pre-created Web App “meantodosample01” in Stage 3.
- In Image Source, select “Built-in Image”.
- In Runtime Stack, obviously, please select Node.js 8.1.

6. Click on “Deploy Azure App Service”, take some modification in “Package or folder”, instead of `$(System.DefaultWorkingDirectory)/**/*.zip`.

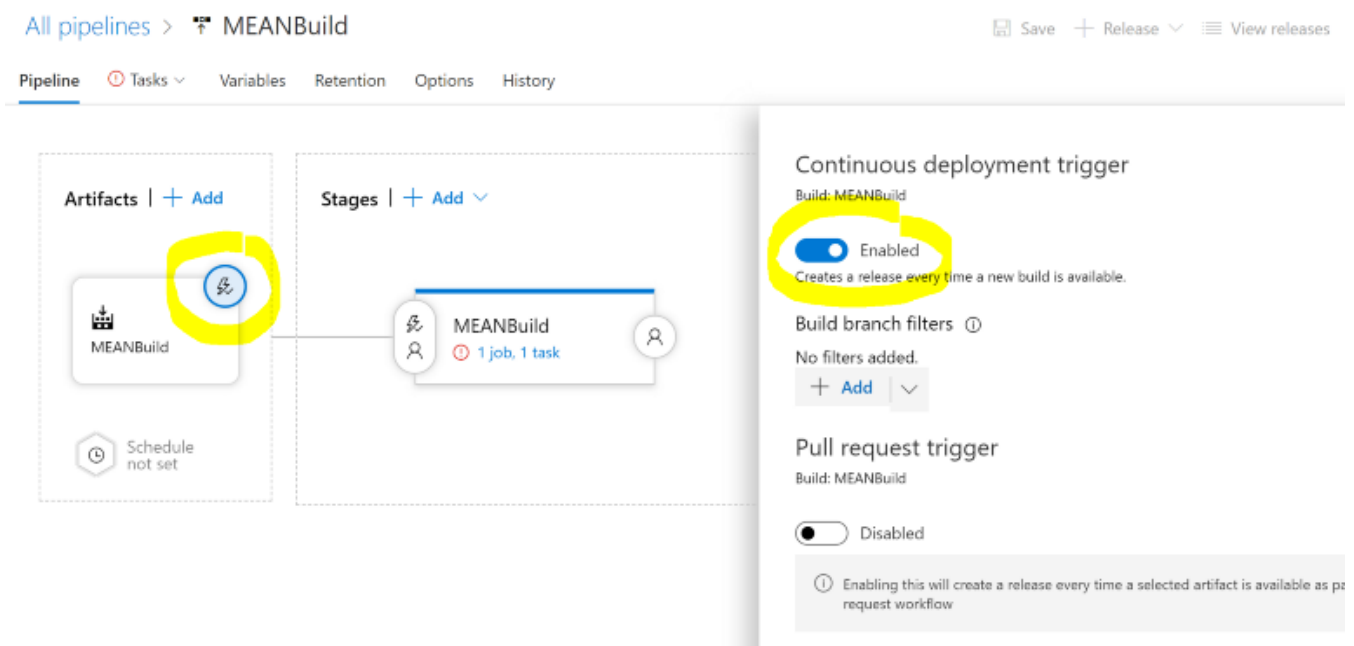
e.g. `$(System.DefaultWorkingDirectory)/MEANBuild/drop/*.zip`

- MEANBuild is your Build Pipeline name

- drop is your pre-defined artifact folder name when you configure the Build Pipeline “Publish artifacts” task.

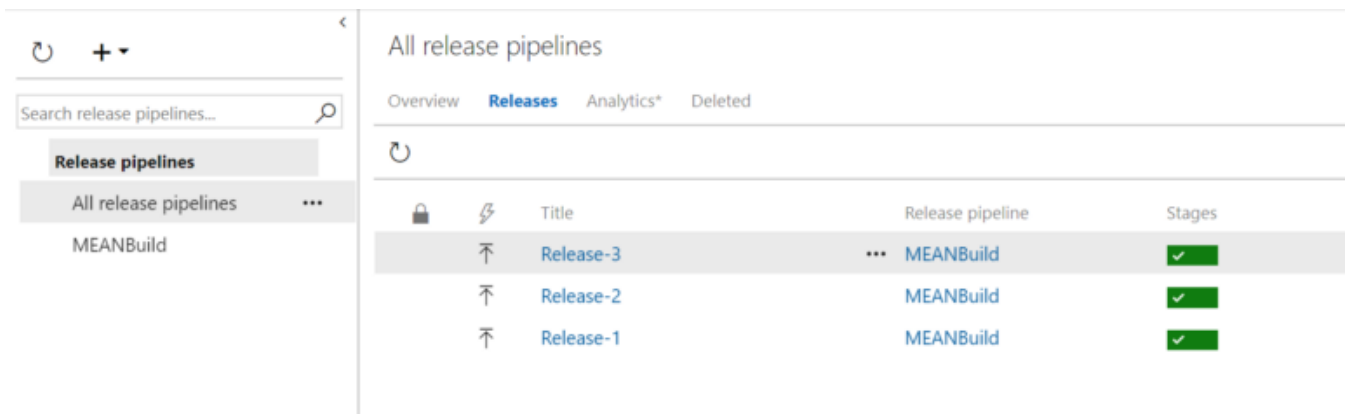


7. Back to top, click “Pipeline” tab, click on the “Electric” icon in Artifacts. Check the “Enabled” box in Continuous deployment trigger. Then, finally, click “Save”.



Up to now, all the Build & Release Pipeline configuration in Azure DevOps has been done. You can either commit some update from local to Azure DevOps Repos to test

end-to-end triggering flow or you can manual trigger Build Pipeline “MEANBuild” then it will trigger the Release Pipeline after build tasks are completed.



The screenshot shows the Azure DevOps 'All release pipelines' page. On the left, a sidebar lists 'Release pipelines' with a sub-item 'All release pipelines' and a specific pipeline named 'MEANBuild'. The main area shows a table of release pipelines. The table has columns for 'Title', 'Release pipeline', and 'Stages'. Three pipelines are listed: 'Release-3', 'Release-2', and 'Release-1', all using the 'MEANBuild' pipeline and showing a green checkmark in the 'Stages' column.

	Title	Release pipeline	Stages
⬆	Release-3	MEANBuild	✓
⬆	Release-2	MEANBuild	✓
⬆	Release-1	MEANBuild	✓

Finally, try to access your Web App URL
(<https://meantodosample01.azurewebsites.net/>).

Update 1 - I'm a Todo-Logger 3

- ☐ i need to buy an apple
- ☐ i need to buy a orange
- ☐ i need to buy a banana

I go to school by bus

Add

This is a demo todo app with MEAN Stack.

Closeout

Hope you're enjoying this article, this article should give you basic understanding for how to make Azure DevOps Build and Deploy Pipeline to work with open-source software stack, MEAN. There are so much interesting Build & Deployment tasks can be added to enrich/enhance your DevOps workflows, e.g. Testing... Having sometime to further explore to make our life easier.

In next article, I'll talk about how to make DevOps working with Data Sciences space...

Related Sharing

- Setup Jenkins Build & Deploy Job with Azure App Service on Linux for Containerized application

[Azure](#) [Nodejs](#) [Angularjs](#) [DevOps](#) [Cloud Computing](#)

[About](#) [Help](#) [Legal](#)

Get the Medium app

