

Angular on Azure — Part I

 Pankaj Parkar [Follow](#) 
May 29, 2019 · 9 min read

Using Azure Pipeline

Developers test software in a local environment using servers such as LAMP, WAMP, node, IIS, and Apache. Local deployment is good in terms of fast development and debugging, but we can't ship our physical machines to the client in order to access to the application 😞. We have to deploy an application to a web server/cloud in order to make it accessible to the end user on their preferred platform (mobile, desktop, etc).

A variety of cloud providers exist in the current market, the most popular being [Microsoft Azure](#), [Google Cloud Platform](#), [AWS](#). These providers offer an unbeatable combination of ease, speed, and automation, so if you have not deployed using such a platform, then this is the article for you! This article focuses on deploying an Angular app to Azure.

What is Deployment?

The action of bringing resources into effective action

In web development, deployment is concerned with making our static/dynamic resources available over the wire so the end user can access them in a desired device, using a browser.

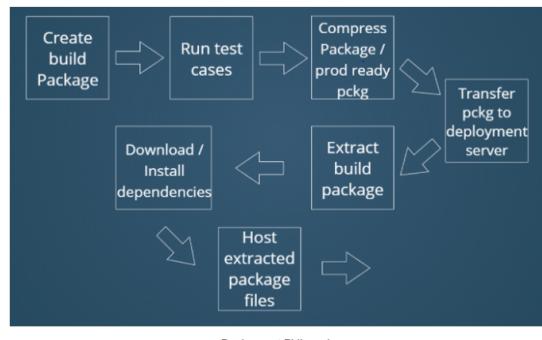
What is hosting?

Web hosting is a service that allows organizations and individuals to post a website or web page onto the Internet.

The deployment process is incomplete without hosting.

Deployment Philosophy

Deployment exposes your web application using a philosophy that has been followed for year. The diagram below outlines typical deployment steps that could be applied to any type of software.



What is Azure?

Azure is a cloud platform service which provides cloud services, including those for compute, analytics, storage, serverless, AI and ML, IoT, containers, DevOps, mobile, and networking. It is widely considered as both a PaaS and IaaS product. This article covers the development, deployment, and DevOps aspects of the platform.

If you are new to Azure, a free subscription is available for those wishing to try

the platform without any commitment.

Azure App Service

The Azure App Service is part of the PaaS section of the platform. It easily builds and deploys a highly available web app to the cloud. Multiple features are available right out of the box, as illustrated below.

The screenshot shows the Microsoft Azure portal interface for creating a new Web App. The sidebar on the left is highlighted with a red box and contains the 'App Services' option, which is circled with number 1. The main window shows the 'Web App' creation wizard. Step 2 is indicated by a red box around the '+ Add' button. Step 3 is indicated by a red box around the 'Resource group' dropdown, which is set to 'pankajparkar-group'. Step 4 is indicated by a red box around the 'Name' field, which is filled with 'angular-deployment'. Step 5 is indicated by a red box around the 'Runtime stack' dropdown, which is set to 'ASP.NET V4.7'. Step 6 is indicated by a red box around the 'Operating System' dropdown, which is set to 'Windows'. Step 7 is indicated by a red box around the 'APP SERVICE PLAN' dropdown, which is set to '(New) ASP-pankajparkar-group'. Step 8 is indicated by a red box around the 'Review and create' button at the bottom. Step 9 is indicated by a red box around the 'Monitoring' link next to it.

The important steps are numbered in the above screenshot. If a resource group is not created, then do so in step 3. Also, if you do not have a service plan, create one at this time. Make sure you select 'ASP .NET 4.7' in the 'Runtime Stack' option in step 5. For more information, follow the guide for the detailed creation of [Azure Service Plan](#) and [Azure App Service](#).

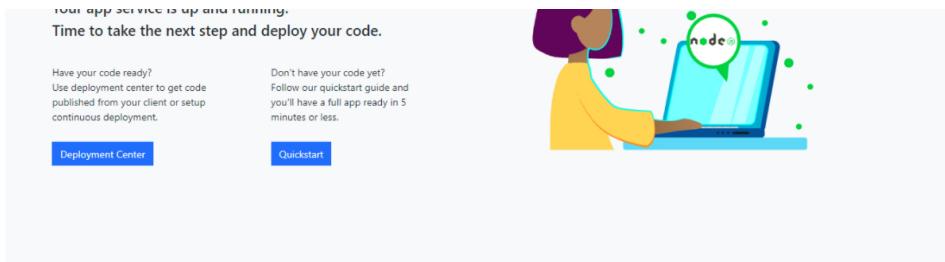
Once you're done with the fill in details, click on "Review and create" button and then on the next screen press "Create" button. To see the newly created resource you can click on "All Resources" option in the sidebar.

The screenshot shows the Microsoft Azure portal's 'All resources' page. The sidebar on the left is highlighted with a red box and contains the 'All resources' option, which is circled with number 1. The main table lists four resources: 'angular-deployment' (App Service), 'ASP-pankajparkar-group-a734' (App Service plan), 'common-appserviceplan' (App Service plan), and 'pankajparkar' (App Service). The 'angular-deployment' row is highlighted with a red box and circled with number 2. The 'LOCATION' column for all resources is 'Central US'. A message at the bottom of the table says 'Newly created resource group added'.

The following url can be loaded to check if the recently deployed application is available in the cloud, <https://<app-name>.azurewebsites.net/>

In my case I used app name as "angular-deployment" so URL would become <https://angular-deployment.azurewebsites.net/>

The screenshot shows a web browser displaying the deployed Angular application. The address bar shows the URL <https://angular-deployment.azurewebsites.net/>. The page content includes the Microsoft Azure logo and the text 'Hey, Node developers! Your app service is up and running.' There is also a small circular profile picture in the bottom right corner.



But, before moving forward, we should minimize the final bundle size of the application. No worries; that process is discussed in a later section.

Make angular application production ready

Angular CLI tooling is incredible; simply executing `ng serve` compiles angular code to Javascript and generates bundle files. For a simple hello-world app, however, the total file size is far short of desirable.

Angular currently offers two compilation modes

1. Just in Time (JIT) Mode
2. Ahead of Time (AOT) Mode

In short, JIT ships the Angular compiler over the wire and component templates are compiled inside the browser. AOT mode precompiles all templates and makes the resulting JS available for further optimization before shipping the bundled application over the wire. Smaller application sizes and quicker response makes for better UX!

For those new to the Angular CLI, AOT mode is enabled with the command

```
ng build --prod
```

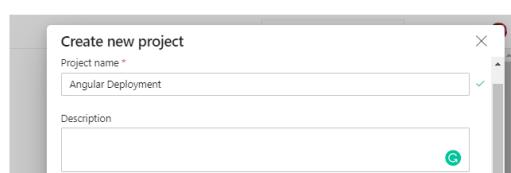
This command compiles all templates, then applies tree-shaking, optimization, minification, and bundling to create a final, highly-optimized package. All distribution files are automatically placed in the `dist` folder of your project, which can be directly hosted to a cloud provider.

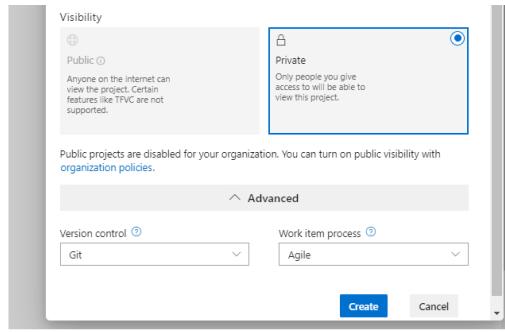
Setup Deployment

In this article, [Azure DevOps](#) (formerly known as VSTS) is used to deploy an application to the cloud.

The screenshot shows the Azure DevOps interface. On the left, there's a sidebar with 'My organizations' and a red box around the 'pankajparkar' organization. In the main area, there's a list of projects with 'weather-app' selected, indicated by a red box. At the top right, there's a '+ Create project' button with a red box around it. Below the projects list, there's a 'DevOps create project' dialog box.

If you have not created an organization, then do so before clicking the 'Create Project' button, as shown in the above diagram. This displays the 'Create New Project' dialog.





In “Create new project” screen, fill project name, description and select visibility (I selected private). Also, version control was set to ‘Git’, and ‘Work item process’ defaulted to ‘Agile.’ Then, click the ‘Create’ button.

The dashboard page is displayed after a project is created. Several actions may be performed from the dashboard sidebar.

Azure DevOps Gist

1. Boards — To track down tasks and progress of a project
2. Repos — Code repositories involved in the development.
3. Pipelines — CI / CD pipeline
4. Test Plans — Provides all tools to successfully test your applications
5. Artifacts — Provides a secure, highly performant store and easy feed

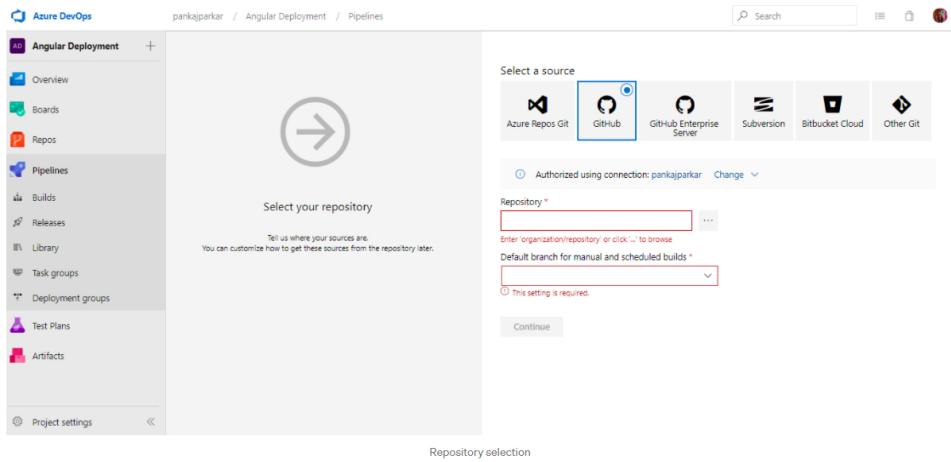
The most important feature in the above list for purposes of this article is the Azure Pipelines setup.

Create A Brand New Pipeline

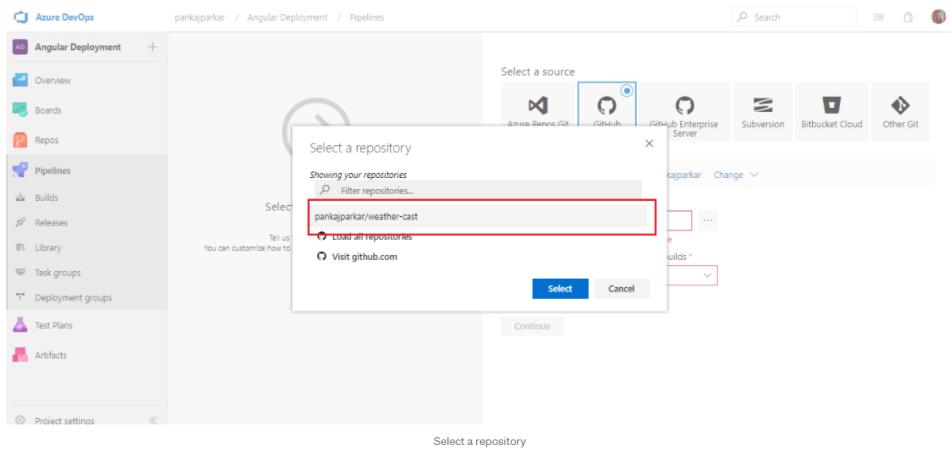
Select the ‘Pipelines’ option from the left sidebar, which displays the ‘New Pipeline’ button in the middle of the screen. The following dialog is displayed after clicking the ‘New Pipeline’ button.



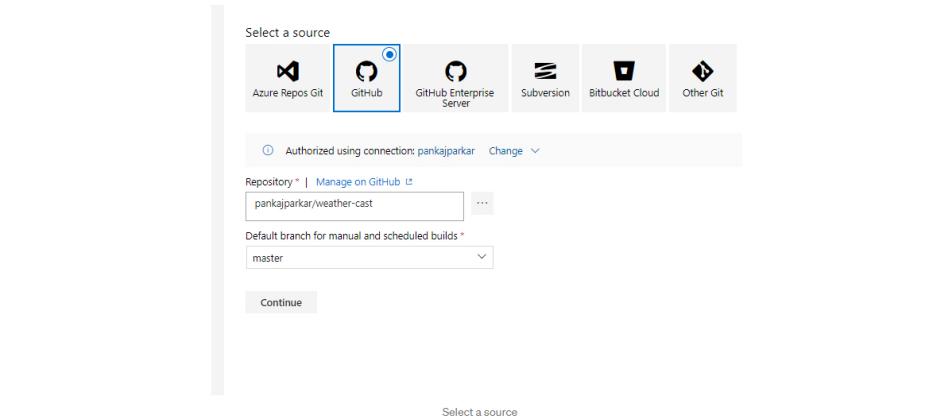
Pipelines are created with `yaml` files. A new `yaml` file may be created with a visual tool or by using the 'Use the classic editor' link at the bottom of the dialog.



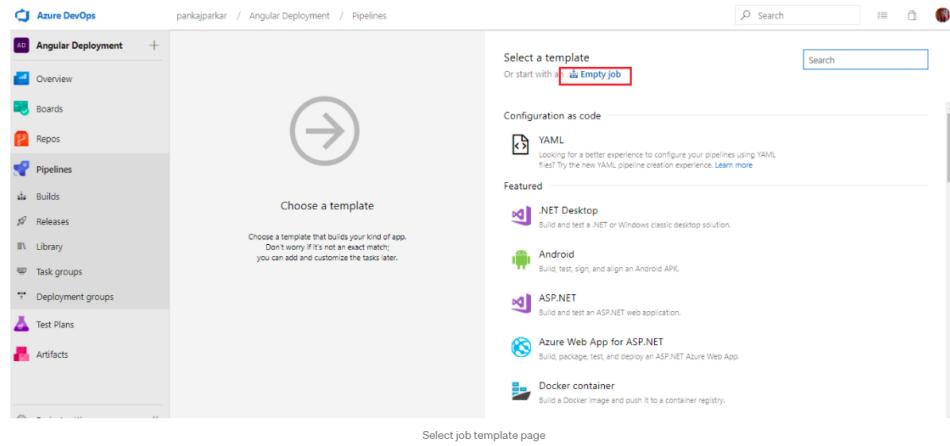
The next step is selecting a repository resource, which can be a new repository
(above) or using an existing repo as shown below. I'm using my existing
Github repo, so I selected 'Github' at this stage. To select a Github repo,
click on the '...' Button to browse repositories.



Select the desired repository for deployment. In this example, I selected the
'weather-cast' repo. Then, click the 'Select' button.



At this point, you are very close to creating a brand new pipeline! By
default, the 'master' branch is selected. Click on the 'Continue' button.



Now, you've made it to the final page of pipeline creation! Next, we create a 'Job,' or the steps involved in the actual deployment. For now, just select 'Empty Job' to create a blank Job with no content. Don't worry, we will add steps for it in the next section.

After the pipeline is created, you will see a screen where an Agent pool is assigned to run a job when any tasks are to be deployed. For this tutorial, we are going to setup deployment tasks under the 'Agent Job 1.' Simply click on the '+' button in the dialog.

Setting Up Pipeline Step

Cool! We've finally made it to the stage where we can add tasks for the deployment job! Refer to the following screen shot.

1. Install node

After clicking the '+' icon beside 'Agent Job 1,' you can search by 'node' in

the list (item 1 in the screen shot) then select ‘Node Tool Installer.’ When that dialog displays (item 2), click the ‘Add’ button (item 3 in the above screenshot).

This screenshot shows the Azure DevOps Pipelines interface for a project named 'Angular Deployment'. On the left sidebar, under 'Pipelines', the 'Builds' tab is selected. In the main pane, a pipeline named 'Angular Deployment-Cl' is displayed. The pipeline consists of two agent jobs: 'Agent job 1' and 'Agent job 2'. 'Agent job 1' has one task: 'Node Available on machine'. 'Agent job 2' has one task: 'Node.js Tool Installer'. A red circle labeled '1' highlights the 'Node.js Tool Installer' task. A red circle labeled '2' highlights the 'Display name' field, which contains 'Node Available on machine'. A red circle labeled '3' highlights the 'Version Spec' field, which contains '8.x'. Below the task list, a message says 'Making node available on machine'.

This displays the first task in the ‘Agent job 1’ list. Next, fill in the details for this task. Enter display and version spec, as shown above. This configures NodeJS on our VM.

2. Install Angular CLI

This screenshot shows the continuation of the pipeline configuration. The 'Agent job 1' section now includes an additional task: 'Install Angular CLI'. A red circle labeled '1' highlights this new task. A red circle labeled '2' highlights the 'Display name' field, which contains 'Install Angular CLI'. A red circle labeled '3' highlights the 'Command' dropdown, which is set to 'custom'. A red circle labeled '4' highlights the 'Command and arguments' field, which contains 'install -g @angular/cli'. Below the task list, a message says 'Install Angular CLI to machine'.

As before, search for ‘npm’ in the task list and then click the ‘Add’ button. Fill in the details as shown above to install the Angular CLI as the next step in the task list.

3. npm install

This screenshot shows the final step in the pipeline configuration. The 'Agent job 1' section now includes three tasks: 'Node Available on machine', 'Install Angular CLI', and 'npm install'. A red circle labeled '1' highlights the 'npm install' task. A red circle labeled '2' highlights the 'Display name' field, which contains 'npm install'. A red circle labeled '3' highlights the 'Command' dropdown, which is set to 'install'. Below the task list, a message says 'npm install on package folder'.

Continue the same process as above to create a task that installs all npm dependencies.

4. Create Prod Build

The screenshot shows the Azure DevOps Pipelines interface for the 'Angular Deployment-CI' pipeline. On the left sidebar, under the 'Pipelines' section, the 'Create Prod Build' task is highlighted with a red circle labeled '1'. The main pane displays the task configuration:

- Display name:** Create Prod Build (circled with '2')
- Command:** custom (circled with '3')
- Working folder that contains package.json:** (empty input field)
- Command and arguments:** run build (circled with '4')
- Custom registries and authentication:** (disabled)
- Control Options:** (disabled)
- Output Variables:** (disabled)

Again add `npm` ask and fill in the details shown above. This time select command as in “custom”, and “command and arguments” would be `run build`. Basically, it calls `ng build --prod` command written as scripts in `scripts` section of the `package.json` file.

This is the task that creates the production-ready Angular package. Continue as before using the details shown in the above screenshot. ‘Command’ is ‘custom’ and the ‘Command and arguments’ input is ‘`ng build — prod`’. This causes the `ng build --prod` command to be written in the ‘scripts’ section of the `package.json` file.

5. Host dist folder

Next, search for ‘Azure App Service Deploy’ and add it to the task list. Fill in the details as shown below. This task hosts and deploys code to the server.

The screenshot shows the Azure DevOps Pipelines interface for the 'Angular Deployment-CI' pipeline. The 'Azure App Service Deploy: angular-deployment' task is highlighted with a red circle labeled '1'. The main pane displays the task configuration:

- Display name:** Azure App Service Deploy: angular-deployment (circled with '2')
- Connection type:** Azure Resource Manager (circled with '3')
- Azure subscription:** Visual Studio Enterprise (circled with '4')
- App Service type:** Web App on Windows (circled with '5')
- App Service name:** angular-deployment (circled with '6')
- Package or folder:** dist/ (circled with '7')

After you have finished entering details, click on the ‘Save and queue’ button. This saves and subsequently runs the pipeline. You will see a message with a hyperlink containing a build number. Or, you may navigate to ‘Pipelines > Builds’ to see the following screen.

The screenshot shows the Azure DevOps Pipelines interface for the 'Angular Deployment-CI' pipeline. The build log for job #54 is displayed, showing the following steps:

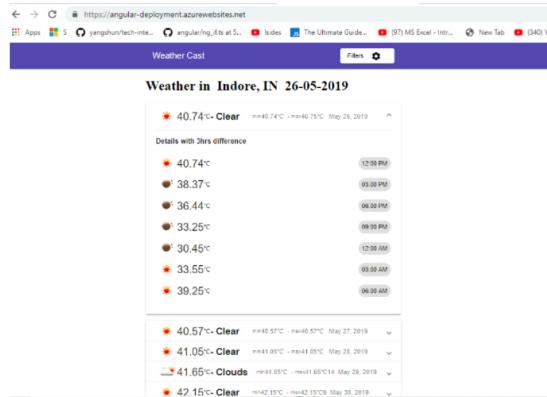
- Prepare job - succeeded
- Initialize job - succeeded
- Checkout - succeeded
- Node Available on Machine - succeeded

The log indicates the build started at 5/26/2019, 2:47:50 PM and took 4m 55s.

Deployment groups		Install Angular CLI · succeeded	1m 3s
Test Plans		npm install · succeeded	1m 47s
Artifacts		Create Prod Build · succeeded	1m 14s
		Install in dist folder · succeeded	6s
		Azure App Service Deploy: angular-deployment · succeeded	32s
Project settings		Post-job: Checkout · succeeded	<1s

After the build is finished

After the job is finished, we can check it as shown below.



Final Azure Pipeline

Tasks Variables Triggers Options Retention History | Save & quit

Pipeline
build pipeline

Get sources
pamaparkar/weather-cast master

Agent job 1
Run on agent

- Node available on machine
- Install Angular CLI
- npm install
- Create Prod Build

Azure App Service Deploy: angular-deployment

Final pipeline tasks

Conclusion

This article outlined the steps to deploy an Angular application to Azure directly from Github or another repository. An Azure pipeline is a powerful feature to setup and visualize a deployment job with minimal effort.

If you like this article press clap button 50 times or as many times you want. Feel free to ask a question if you have any. Thanks a lot for reading!

Soon, I will release part II of this article which covers CI and CD with Azure Pipeline. Stay tuned!

For more Angular goodness, be sure to check out the latest episode of [The Angular Show podcast](#).

EnterpriseNG is coming November 4th & 5th, 2021.

Come hear top community speakers, experts, leaders, and the Angular team present for 2 stacked days on everything you need to make the most of Angular in your enterprise applications.

Topics will be focused on the following four areas:

- Monorepos
- Micro frontends
- Performance & Scalability
- Maintenance & Evolution

Sign up for the ng-conf Newsletter
By ngconf
Get up-to-date info, news, special offers and more from ng-conf! Take a look.

Your email 

By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.

Azure Angular DevOps Deployment Azure Pipelines

2.1K Q 6



WRITTEN BY

Pankaj Parkar

 Virtusa | MVP | GDE | Tech Savvy



ngconf

The World's Best Angular Conference



More From Medium

How to Integrate Sign In with Apple in your Application?



Naveen Sharma

I chose coding bootcamp.



Mikayla Toffler

Approaching a Problem...



Usama Rehan

CS373: Week 6



Ethan Lao in Trees Grow Down

Container Orchestration – Technology Choices For Microservices and Other Workloads



Michael Douglass in codeburst

Add ARRIVE to Your iOS App in Just 6 Steps



Rakuten Ready in Rakuten Ready



Yallo, a new Tezos language

Davide Gessa (dakk) in Coinmonks

Learn more.

Medium is an open platform where 170 million readers come to find insightful and dynamic thinking. Here, expert and undiscovered voices alike dive into the heart of any topic and bring new ideas to the surface. [Learn more](#)

Make Medium yours.

Follow the writers, publications, and topics that matter to you, and you'll see them on your homepage and in your inbox. [Explore](#)

Write a story on Medium.

If you have a story to tell, knowledge to share, or a perspective to offer — welcome home. It's easy and free to post your thinking on any topic. [Start a blog](#)



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