

Packaging and Deploying SharePoint Framework Solutions



Agenda

- Creating the App Catalog Site in SharePoint Online
- Packaging SPFX Solutions for Production
- Packaging JavaScript Libraries as External References
- Customizing SPFX Builds with Gulp and Webpack
- Publishing and Updating SPFX Solutions
- Installing an SPFX Solution in a SharePoint Site



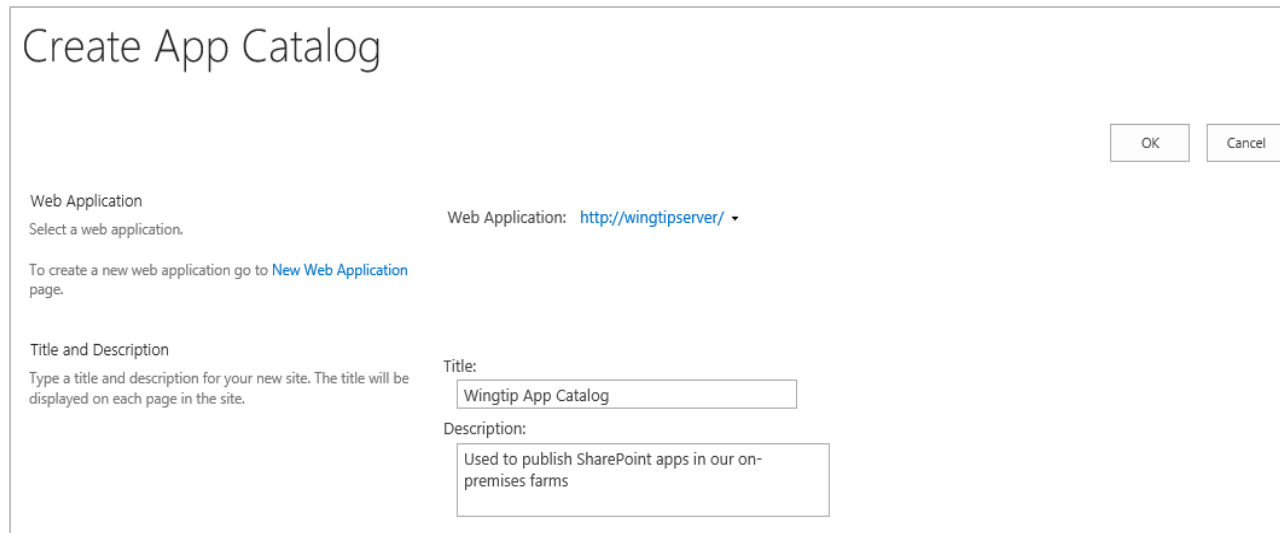
Understanding the App Catalog

- App publishing scheme based on App Catalog
 - App Catalog is site collection with special doc library
 - App packages are published (uploaded) to app catalog
 - Provides better app discovery, installation and upgrade
- App Catalog in on-premises farms
 - One App Catalog site required for each web application
 - End users often play role of App Catalog administrator
- App Catalog in Office 365 & SharePoint Online
 - One App Catalog site used to manage entire tenancy



Creating the App Catalog Site Collection

- You must create the App Catalog site collection
 - You can create it using a PowerShell script
 - You can create it using Central Administration
 - App Catalog site associated with one web application



The screenshot shows the 'Create App Catalog' dialog box. It has a title bar 'Create App Catalog' and 'OK' and 'Cancel' buttons in the top right corner. The dialog is divided into two main sections. The top section is titled 'Web Application' and contains the text 'Select a web application.' followed by 'Web Application: <http://wingtipserver/>'. Below this, it says 'To create a new web application go to [New Web Application](#) page.' The bottom section is titled 'Title and Description' and contains the text 'Type a title and description for your new site. The title will be displayed on each page in the site.' It has two input fields: 'Title:' with the value 'Wingtip App Catalog' and 'Description:' with the value 'Used to publish SharePoint apps in our on-premises farms'.

Create App Catalog

OK Cancel

Web Application

Select a web application.

Web Application: <http://wingtipserver/>

To create a new web application go to [New Web Application](#) page.

Title and Description

Type a title and description for your new site. The title will be displayed on each page in the site.

Title:

Wingtip App Catalog

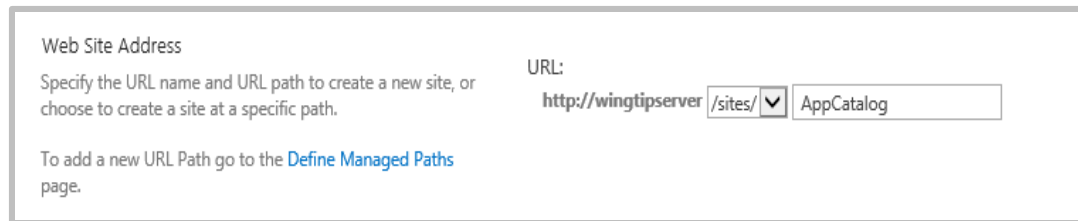
Description:

Used to publish SharePoint apps in our on-premises farms



App Catalog URL and Permissions

- App catalog site created at a specific URL
 - Creating App Catalog site with PowerShell is more flexible
you can create site as top-level domain using host-named site collections (HNSCs)



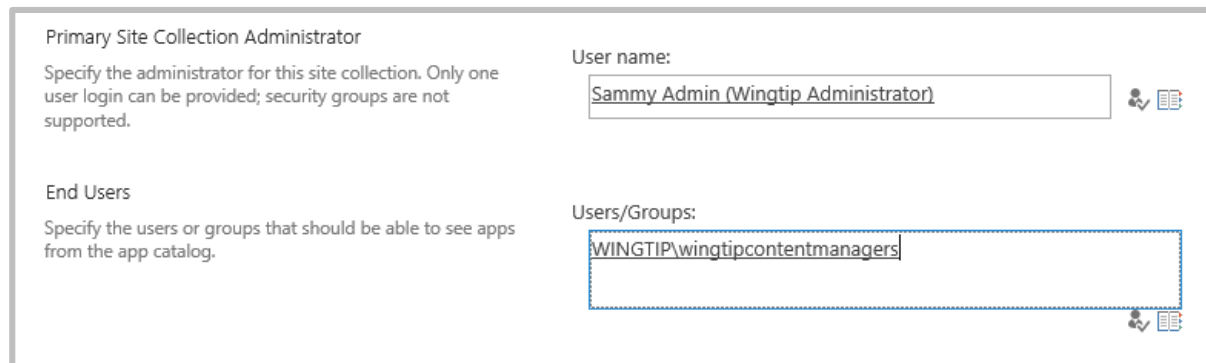
Web Site Address

Specify the URL name and URL path to create a new site, or choose to create a site at a specific path.

URL:

To add a new URL Path go to the [Define Managed Paths](#) page.

- Setting App Catalog permissions
 - Site collection administrator becomes App Catalog administrator
 - End user permissions allows user to discover and install apps



Primary Site Collection Administrator

Specify the administrator for this site collection. Only one user login can be provided; security groups are not supported.

User name:

End Users

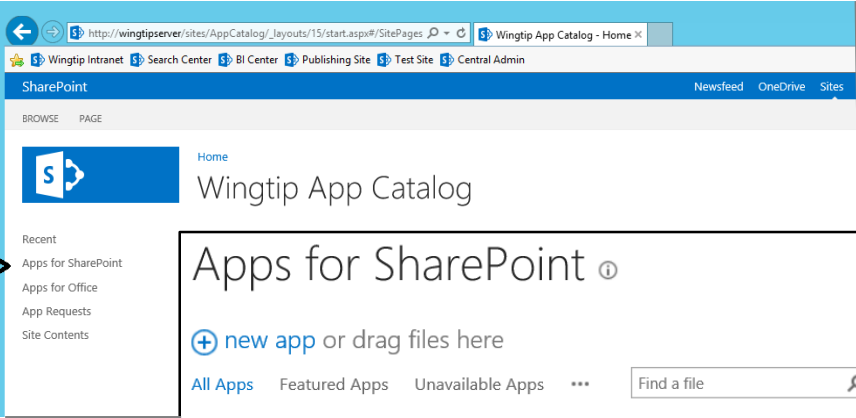
Specify the users or groups that should be able to see apps from the app catalog.

Users/Groups:







Apps for SharePoint Document Library

- Apps for SharePoint is special document library
 - It's the place where you publish SharePoint apps
 - You upload app package and enter the related metadata



The screenshot shows the Wingtip App Catalog interface. The left sidebar contains a 'Recent' section with links to 'Apps for SharePoint', 'Apps for Office', 'App Requests', and 'Site Contents'. An arrow points from 'Apps for SharePoint' to the main content area. The main content area is titled 'Apps for SharePoint' and includes a '+ new app or drag files here' button. Below this is a search bar and a table of installed apps.

✓	📄	Title	Name	App Version	Edit	Product ID	Metadata Language	Default Metadata Language
Product ID : {921FDA40-576C-48CF-B00B-7285D24372A0} (1)								
	📄	Wingtip Search App	WingtipSearchApp 	1.0.0.0		{921FDA40-576C-48CF-B00B-7285D24372A0}	English - 1033	Yes
Product ID : {2AB41D1E-C8A8-400A-AD6E-7C15FEE6A69A} (1)								
	📄	Calculator App	Calculator 	2.0.0.0		{2AB41D1E-C8A8-400A-AD6E-7C15FEE6A69A}	English - 1033	Yes

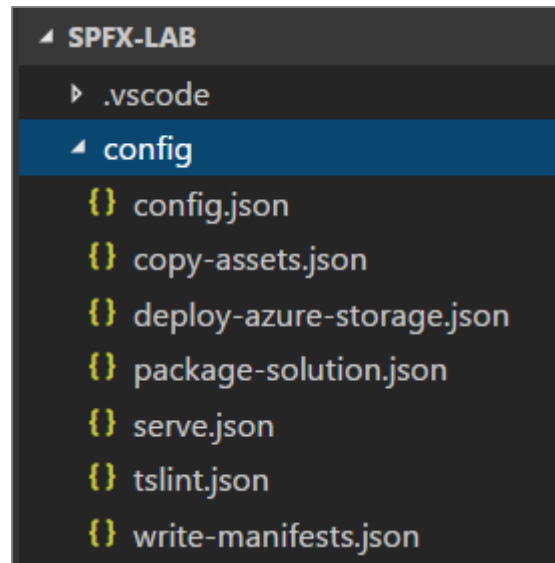
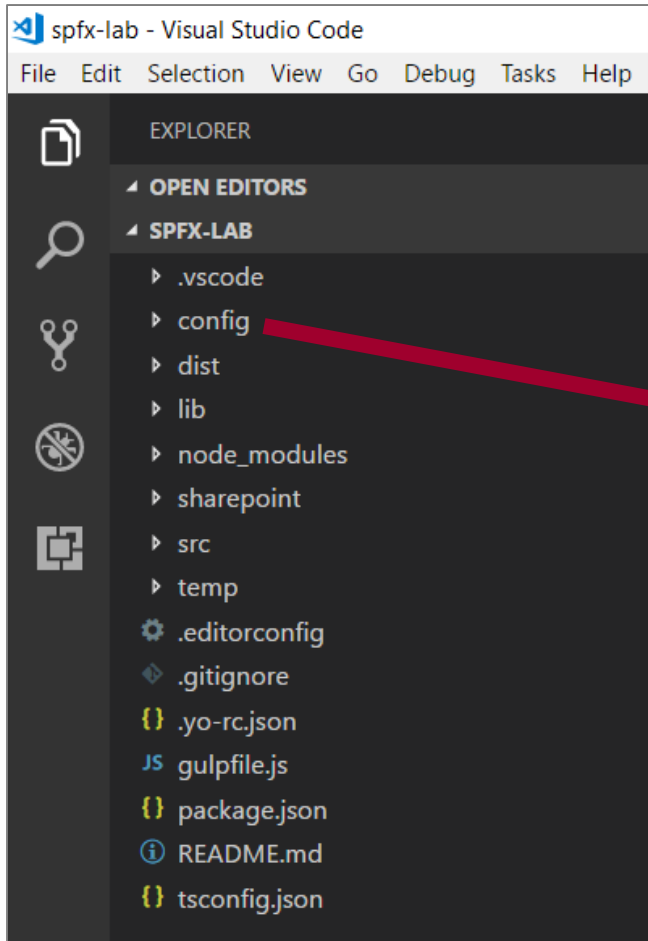




DEMO

Creating an App Catalog Site

SPFx Project Configuration Files



package-solution.json

- package-solution.json
 - Contains top-level project properties (`id`, `name`, `version`)
 - `includeClientSideAssets`
 - `skipFeatureDeployment`
 - `zippedPackage`

```
{  
  "package-solution.json"  
}  
  
1 {  
2   "$schema": "https://dev.office.com/json-schemas/spfx-build/package-solution.schema.json",  
3   "solution": {  
4     "name": "SPFX Lab",  
5     "id": "aa5d865d-3cfb-4fd6-a4d4-6fc5d21dd9d7",  
6     "version": "1.0.0.0",  
7     "includeClientSideAssets": true,  
8     "skipFeatureDeployment": true  
9   },  
10  "paths": {  
11    "zippedPackage": "solution/spfx-lab.sppkg"  
12  }  
13 }
```

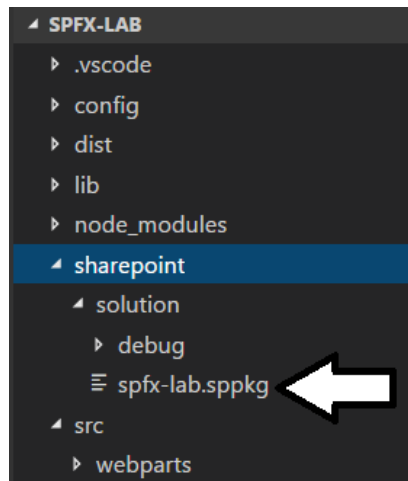


Building a SPFx Solution

- Done by executing gulp package-solution
 - Can be done with or without **--ship** parameter

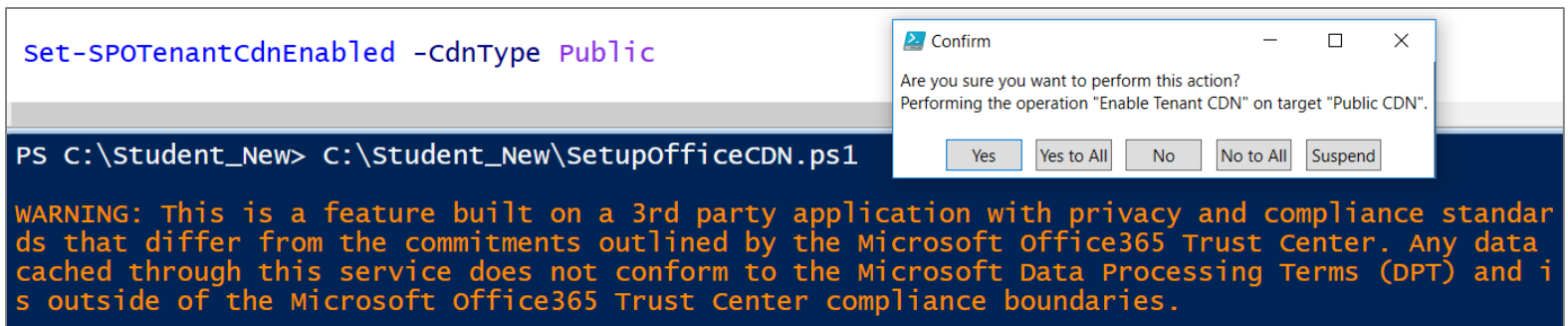
```
PS C:\Demos\spfx-lab> gulp package-solution
Build target: DEBUG
[20:02:30] Using gulpfile C:\Demos\spfx-lab\gulpfile.js
[20:02:30] Starting gulp
[20:02:30] Starting 'package-solution'...
[20:02:30] Starting subtask 'configure-sp-build-rig'...
[20:02:30] Finished subtask 'configure-sp-build-rig' after 4.95 ms
[20:02:30] Starting subtask 'package-solution'...
[20:02:30] Warning - [package-solution] This is not a production build (--ship or --production).
```

- Generates zip archive with **.sppkg** extension



Enabling the Office 365 CDN

- Office 365 provides CDN for SPFx solution deployment
 - Office 365 CDN must be enabled using PowerShell



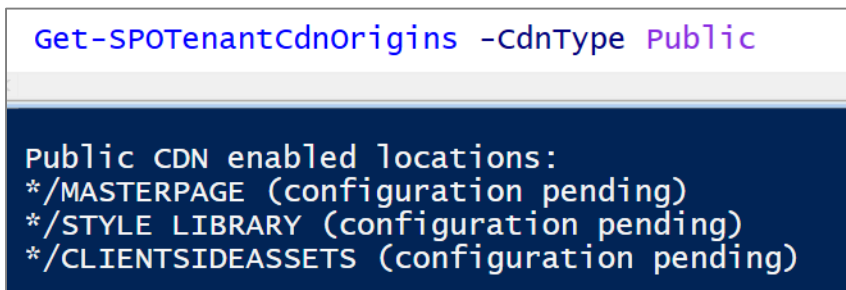
The screenshot shows a PowerShell terminal window with the command `Set-SPOTenantCdnEnabled -cdnType Public` entered. Below the command, a warning message is displayed: "WARNING: This is a feature built on a 3rd party application with privacy and compliance standards that differ from the commitments outlined by the Microsoft Office365 Trust Center. Any data cached through this service does not conform to the Microsoft Data Processing Terms (DPT) and is outside of the Microsoft Office365 Trust Center compliance boundaries." Overlaid on the terminal is a "Confirm" dialog box asking "Are you sure you want to perform this action?" and "Performing the operation 'Enable Tenant CDN' on target 'Public CDN'." The dialog box has buttons for "Yes", "Yes to All", "No", "No to All", and "Suspend".

```
Set-SPOTenantCdnEnabled -cdnType Public
```

PS C:\Student_New> C:\Student_New\SetupOfficeCDN.ps1

WARNING: This is a feature built on a 3rd party application with privacy and compliance standards that differ from the commitments outlined by the Microsoft Office365 Trust Center. Any data cached through this service does not conform to the Microsoft Data Processing Terms (DPT) and is outside of the Microsoft Office365 Trust Center compliance boundaries.

- Enabling CDN creates `*/CLIENTSIDEASSETS` origin



The screenshot shows a PowerShell terminal window with the command `Get-SPOTenantCdnOrigins -cdnType Public` entered. The output of the command is displayed below: "Public CDN enabled locations: */MASTERPAGE (configuration pending) */STYLE LIBRARY (configuration pending) */CLIENTSIDEASSETS (configuration pending)".

```
Get-SPOTenantCdnOrigins -cdnType Public
```

Public CDN enabled locations:
*/MASTERPAGE (configuration pending)
*/STYLE LIBRARY (configuration pending)
*/CLIENTSIDEASSETS (configuration pending)


- CDN supports these file type extensions
CSS, EOT, GIF, ICO, JPEG, JPG, JS, MAP, PNG, SVG, TTF, WOFF



includeClientSideAssets = True

- If Office 365 CDN is enabled...
 - it will be used automatically with default settings.
- If Office 365 CDN is not enabled...
 - assets will be served from app catalog site collection.

```
{ } package-solution.json •
1  {
2    "$schema": "https://dev.office.com/json-schemas/spfx-build/package-solution.schema.json",
3    "solution": {
4      "name": "SPFX Lab",
5      "id": "aa5d865d-3cfb-4fd6-a4d4-6fc5d21dd9d7",
6      "version": "1.0.0.0",
7      "includeClientSideAssets": true,
8      "skipFeatureDeployment": true
9    },
10   "paths": {
11     "zippedPackage": "solution/spfx-lab.sppkg"
12   }
13 }
```



Packaging a SPFx Solution for Distribution

- `gulp bundle --ship`
 - Uses dynamic URL for assets based on tenant CDN settings

```
PS C:\Demos\spfx-lab> gulp bundle --ship
Build target: SHIP
[20:51:24] Using gulpfile C:\Demos\spfx-lab\gulpfile.js
[20:51:24] Starting gulp
[20:51:24] Starting 'bundle'...
[20:51:24] Starting subtask 'configure-sp-build-rig'...
[20:51:24] Finished subtask 'configure-sp-build-rig' after 4.45 ms
```

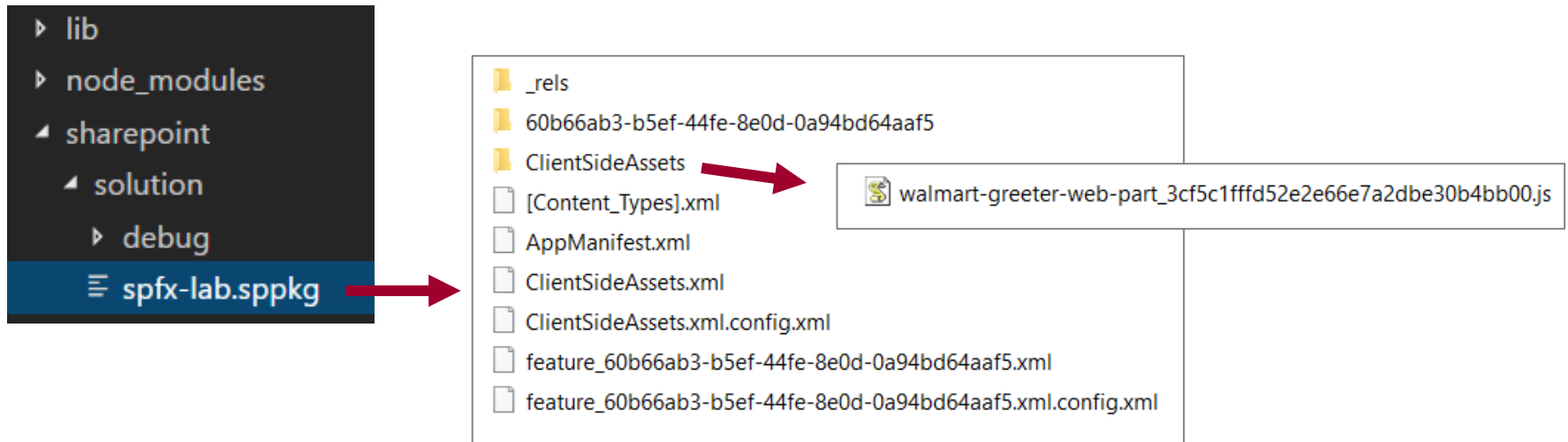
- `gulp package-solution --ship`
 - Uses dynamic URL for assets based on tenant CDN settings

```
PS C:\Demos\spfx-lab> gulp package-solution --ship
Build target: SHIP
[20:53:42] Using gulpfile C:\Demos\spfx-lab\gulpfile.js
[20:53:42] Starting gulp
[20:53:42] Starting 'package-solution'...
[20:53:42] Starting subtask 'configure-sp-build-rig'...
[20:53:42] Finished subtask 'configure-sp-build-rig' after 5.02 ms
[20:53:42] Starting subtask 'package-solution'...
```

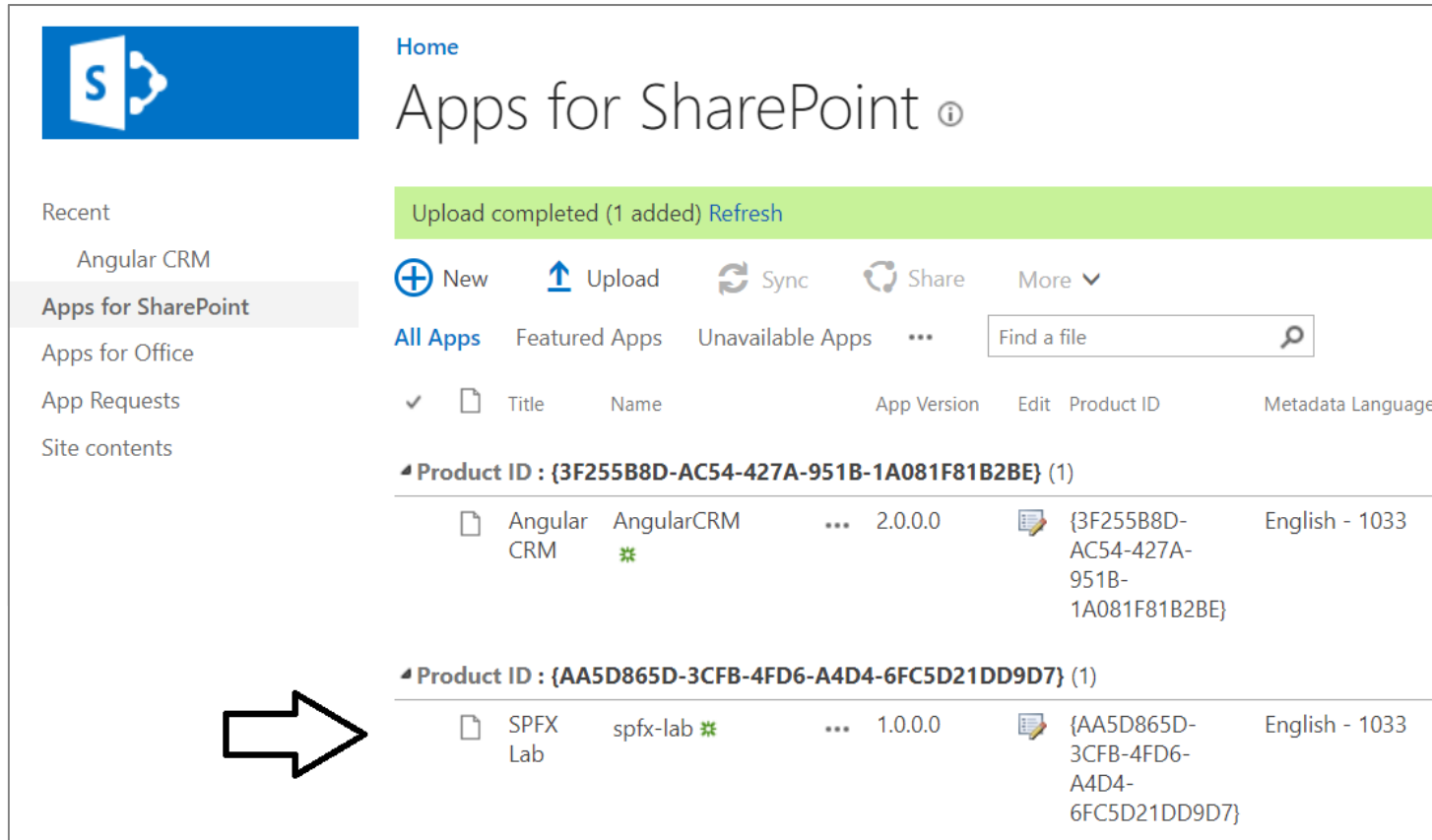


Inside a SPFx Solution Package

- Your .js code is included in ClientSideAssets



Deploying Solution to Office 365 Tenancy



Home

Apps for SharePoint ⓘ

Upload completed (1 added) [Refresh](#)

[New](#) [Upload](#) [Sync](#) [Share](#) [More](#) ▾

[All Apps](#) [Featured Apps](#) [Unavailable Apps](#) ...

✓	📄	Title	Name		App Version	Edit	Product ID	Metadata Language
🔑 Product ID : {3F255B8D-AC54-427A-951B-1A081F81B2BE} (1)								
	📄	Angular CRM	AngularCRM ✨	...	2.0.0.0	✎	{3F255B8D-AC54-427A-951B-1A081F81B2BE}	English - 1033
🔑 Product ID : {AA5D865D-3CFB-4FD6-A4D4-6FC5D21DD9D7} (1)								
	📄	SPFX Lab	spfx-lab ✨	...	1.0.0.0	✎	{AA5D865D-3CFB-4FD6-A4D4-6FC5D21DD9D7}	English - 1033



Agenda

- ✓ Understanding the SharePoint Online App Catalog
- ✓ Publishing and Installing SharePoint Add-ins
- ✓ Packaging SharePoint Framework Projects
- Deploying Provider-hosted Add-ins



Package the web part

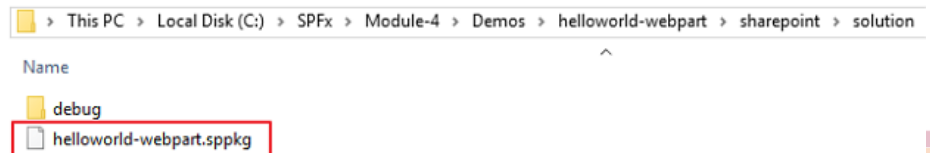
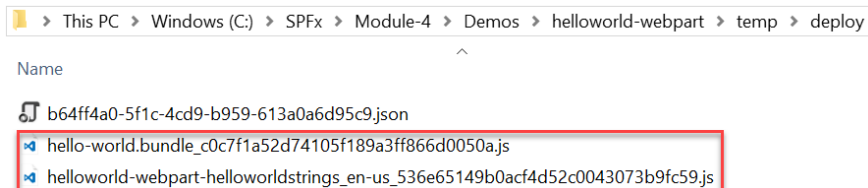
Use the **bundle** gulp task to build, localize, and bundle the project

```
> gulp bundle --ship
```

Use the **package-solution** gulp task to package the project into a .sppkg file

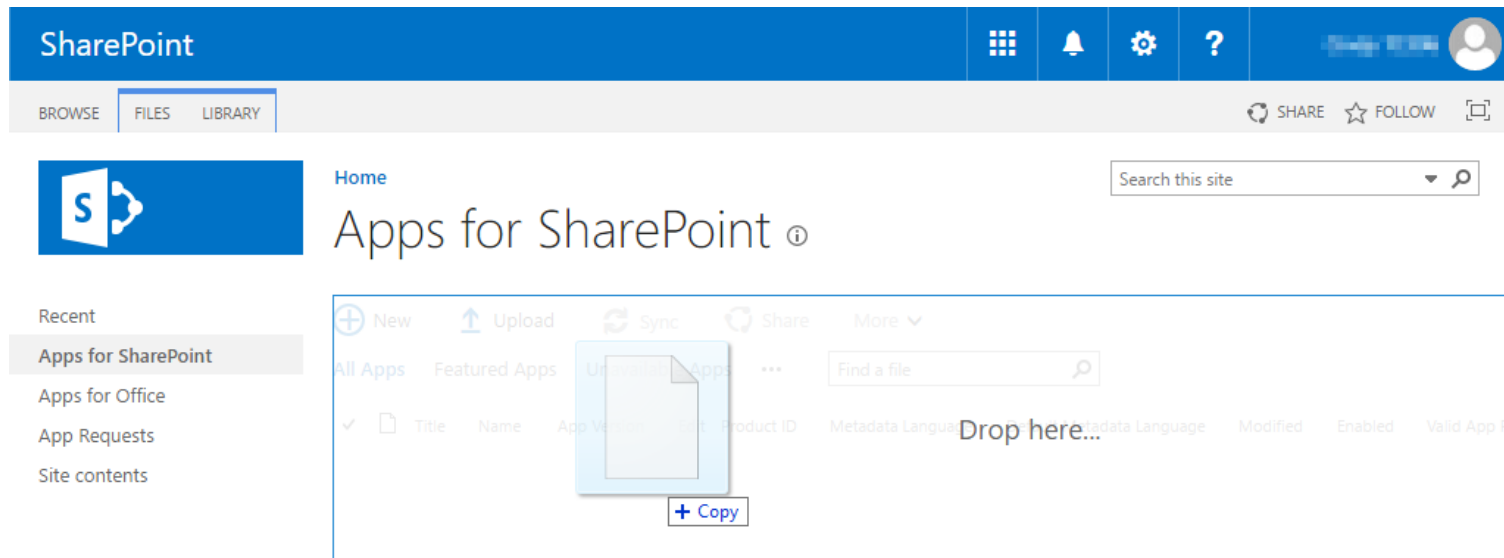
```
> gulp package-solution --ship
```

The **ship parameter** build task creates a minified version of the bundle and copies all of the web part assets, including the web part bundle, into the temp\deploy folder. The .sppkg file is generated in the sharepoint\solution folder.



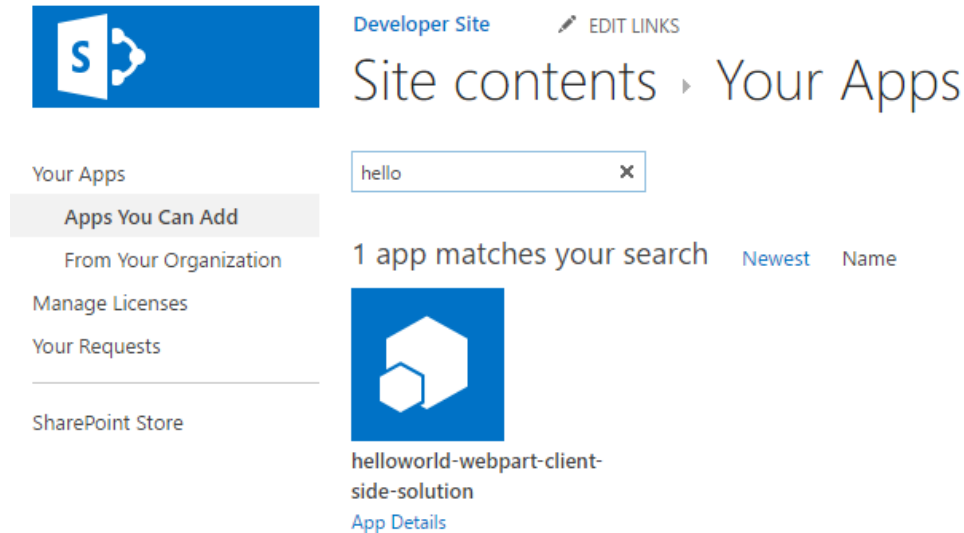
Deploy App to the SharePoint App Catalog

- Go to your Office 365 App Catalog site
- In the left sidebar, choose Apps for SharePoint
- Upload the package (.sppkg file) for the web part



Install the App

- Go to your Office 365 site
- Add the App you just deployed to the SharePoint App Catalog



The screenshot displays the SharePoint App Catalog interface. At the top left is the Office 365 logo. To its right are links for 'Developer Site' and 'EDIT LINKS'. The main heading is 'Site contents ▸ Your Apps'. Below this is a search bar containing the text 'hello'. The results section shows '1 app matches your search' with sorting options 'Newest' and 'Name'. A single app is listed with a blue icon featuring two white hexagons. The app's name is 'helloworld-webpart-client-side-solution' and there is a link for 'App Details'.

Developer Site [EDIT LINKS](#)

Site contents ▸ Your Apps

hello

Your Apps

Apps You Can Add


From Your Organization

Manage Licenses

Your Requests

SharePoint Store

1 app matches your search [Newest](#) [Name](#)



helloworld-webpart-client-side-solution

[App Details](#)



Deploy assets to a CDN and configure web part

- At this point the web part will not run
- You must first
 - deploy the web part assets to a CDN location
 - update the cdnBasePath in the write-manifests.json file

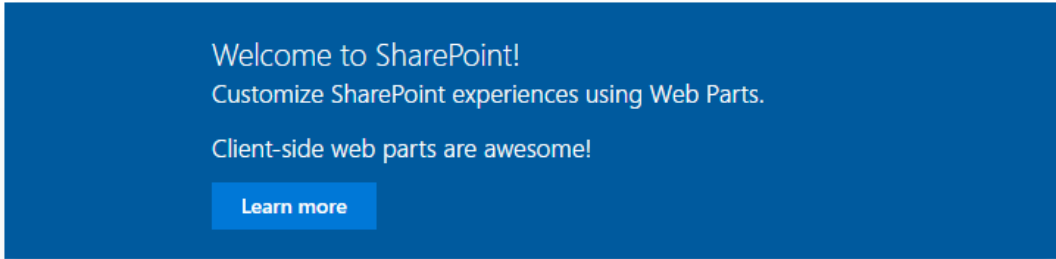
```
{  
  "cdnBasePath": "<!-- PATH TO CDN -->"  
}
```

- See the other sections in this module that describe how to
 - deploy to a SharePoint CDN and an Azure Storage CDN
 - update the cdnBasePath in the write-manifests.json file



Add the web part to a SharePoint page

- After the web part assets are deployed to a CDN location you can preview the web part
- Create a page on your Office 365 site
- Add the web part to the page

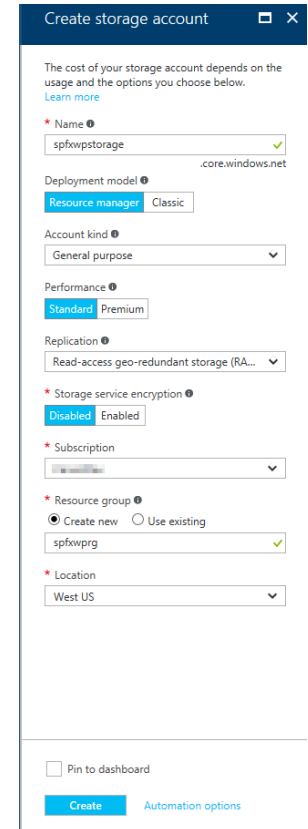
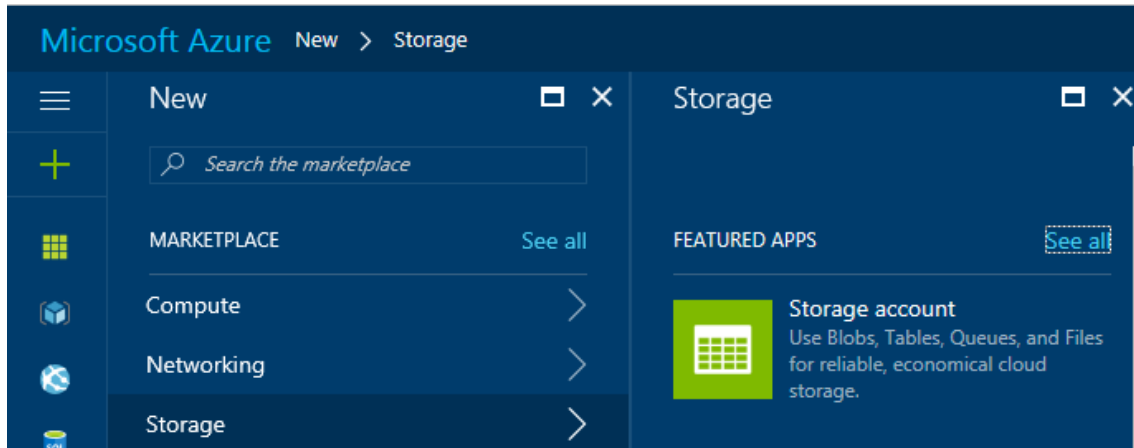
A blue rectangular banner with white text. It contains a welcome message, instructions on customizing SharePoint experiences, and a link to learn more.

Welcome to SharePoint!
Customize SharePoint experiences using Web Parts.
Client-side web parts are awesome!
[Learn more](#)



Create Azure Storage Account

- Go to the Azure Management Portal
- Create a new Storage Account

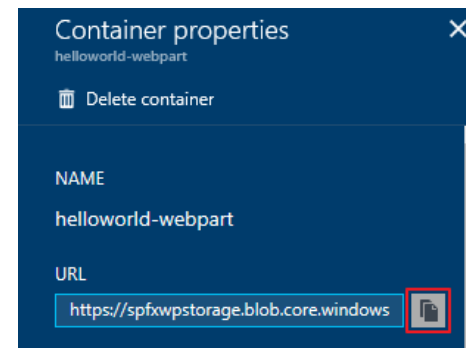
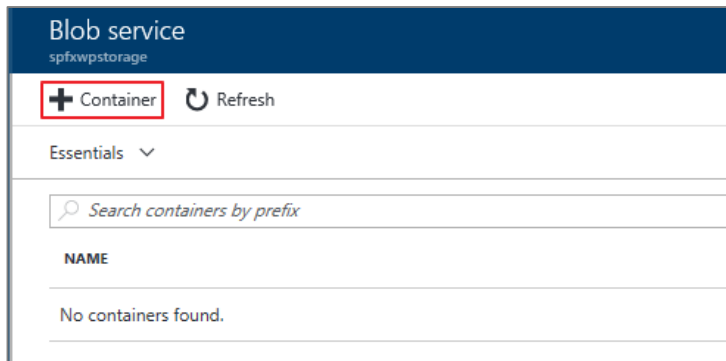
The screenshot displays the 'Create storage account' configuration page. At the top, a note states: 'The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)'. The form includes the following fields and options:

- Name:** A text input field containing 'spfxwpstorage' with a green checkmark icon.
- Deployment model:** Two radio buttons, 'Resource manager' (selected) and 'Classic'.
- Account kind:** A dropdown menu set to 'General purpose'.
- Performance:** Two radio buttons, 'Standard' (selected) and 'Premium'.
- Replication:** A dropdown menu set to 'Read-access geo-redundant storage (RA...'.
- Storage service encryption:** Two radio buttons, 'Disabled' (selected) and 'Enabled'.
- Subscription:** A dropdown menu showing the current subscription.
- Resource group:** Two radio buttons, 'Create new' (selected) and 'Use existing'. Below it, a text input field contains 'spfxwprg' with a green checkmark icon.
- Location:** A dropdown menu set to 'West US'.

At the bottom of the form, there is a checkbox for 'Pin to dashboard' and a blue 'Create' button. To the right of the button is a link for 'Automation options'.

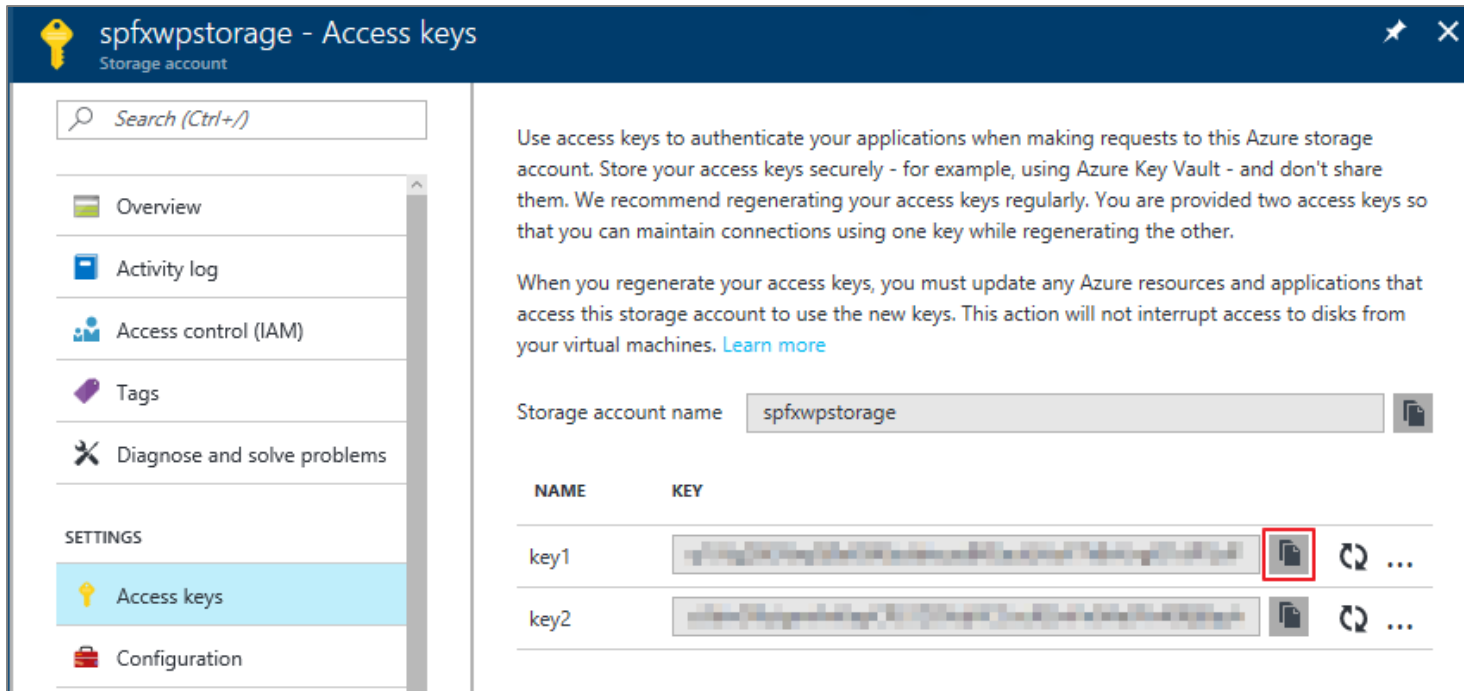
Create Blob Container in Storage Account

- Create a Blob Container and copy the URL



Obtain Storage Account Access Key

- The Storage Account Access Key is used to automate deployments to the Azure Storage CDN, copy one of the keys



The screenshot shows the 'Access keys' page for the storage account 'spfxwpstorage'. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and a SETTINGS section with 'Access keys' (highlighted) and 'Configuration'. The main content area includes instructions on using access keys and a table of the two keys. The first key, 'key1', has its value highlighted with a red box, and a copy icon is visible next to it.

spfxwpstorage - Access keys
Storage account

Search (Ctrl+/,)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

SETTINGS

Access keys

Configuration

Use access keys to authenticate your applications when making requests to this Azure storage account. Store your access keys securely - for example, using Azure Key Vault - and don't share them. We recommend regenerating your access keys regularly. You are provided two access keys so that you can maintain connections using one key while regenerating the other.

When you regenerate your access keys, you must update any Azure resources and applications that access this storage account to use the new keys. This action will not interrupt access to disks from your virtual machines. [Learn more](#)

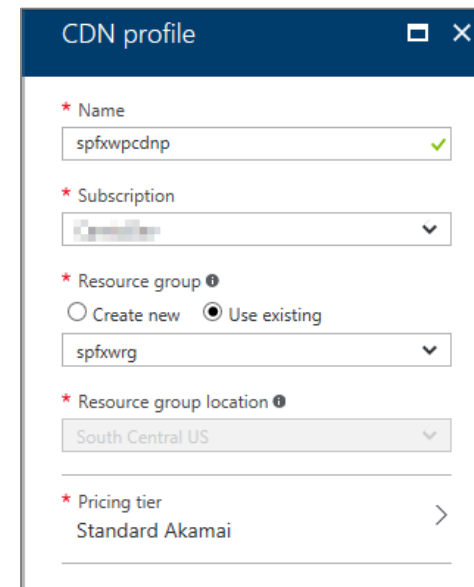
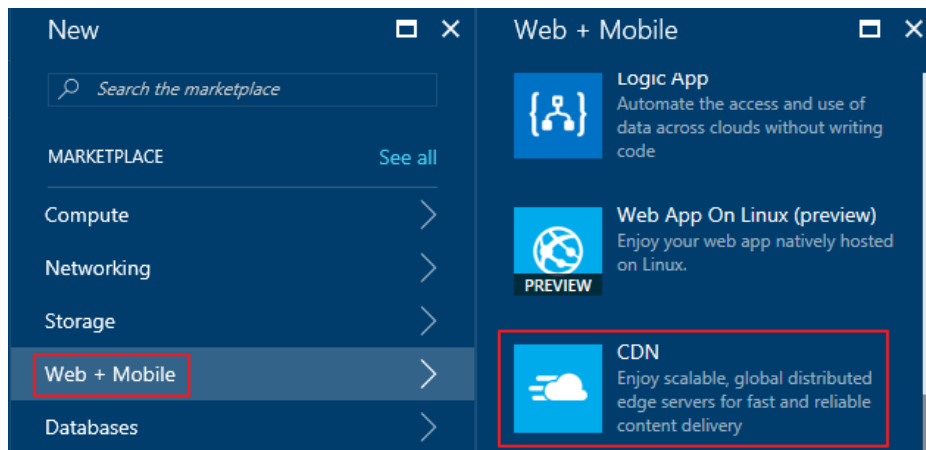
Storage account name: spfxwpstorage

NAME	KEY
key1	[Key Value] [Copy] [Refresh] [More]
key2	[Key Value] [Copy] [Refresh] [More]



Create CDN Profile

- In the Azure Management Portal, create a new CDN Profile



This screenshot shows the 'CDN profile' configuration form. The fields are as follows:

- Name:** spfxwpcdn (with a green checkmark)
- Subscription:** (dropdown menu)
- Resource group:** spfxwrg (with radio buttons for 'Create new' and 'Use existing', where 'Use existing' is selected)
- Resource group location:** South Central US (dropdown menu)
- Pricing tier:** Standard Akamai (with a right arrow)



Create CDN Endpoint

- In the CDN Profile, create a new CDN endpoint

Add an endpoint

Allows configuring content delivery behavior an...

* Name
spfxwpcdne ✓
.azureedge.net

* Origin type
Storage ▼

* Origin hostname ⓘ
spfxwpstorage.blob.core.windows.net ▼

Origin path ⓘ
/Path

Origin host header ⓘ
spfxwpstorage.blob.core.windows.net ✓

Protocol ⓘ	Origin port ⓘ
<input checked="" type="checkbox"/> HTTP	80
<input checked="" type="checkbox"/> HTTPS	443



Configure Webpart to Deploy Assets to CDN

- Update the account, container, and accessKey in the deploy-azure-storage.json file
 - account – Storage account name
 - container – Name of the container you wish to use for the web part
 - accessKey – Storage Account Access Key

```
{  
  "workingDir": "./temp/deploy/",  
  "account": "spfxwpstorage",  
  "container": "helloworld-webpart",  
  "accessKey": "hL2503 ... "  
}
```



Configure the web part to use the CDN

- Create CDN base path

`https://<Storage Account Name>.blob.core.windows.net/<Container Name>`

- Update the `cdnBasePath` in the `write-manifests.json` file

```
{  
  "cdnBasePath": "https://spfxwpstorage.blob.core.windows.net/helloworld-webpart"  
}
```

-



Deploy web part assets to the Azure Storage Account

- Use the deploy-azure-storage gulp task to deploy the assets to the Azure Storage Account

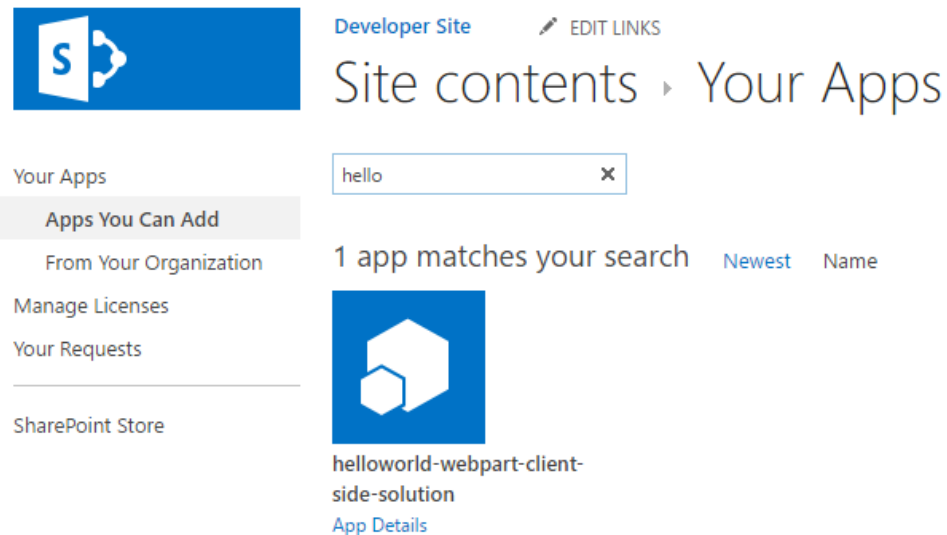
```
> gulp deploy-azure-storage
```

```
> gulp deploy-azure-storage
Build target: DEBUG
[11:56:24] Using gulpfile C:\SPFx\helloworld-webpart\gulpfile.js
[11:56:24] Starting gulp
[11:56:24] Starting 'deploy-azure-storage'...
[11:56:24] Starting subtask 'deploy-azure-storage'...
[11:56:24] [deploy-azure-storage] Uploading files '**/*.*' from directory './temp/deploy/' to Azure
[11:56:25] [deploy-azure-storage] Created container: helloworld-webpart
[11:56:25] [deploy-azure-storage] Uploading [3] files...
[11:56:26] [deploy-azure-storage] Uploaded file: helloworld-webpart-helloworldstrings_en-us_536e65149b0acf4d52c0043073b9fc59.js
[11:56:26] [deploy-azure-storage] Uploaded file: hello-world.bundle_b8a80975dedeb31de300b580fab61182.js
[11:56:26] [deploy-azure-storage] Uploaded file: dd331d09-a9cd-448d-a687-7e43060191e2.json
[11:56:26] [deploy-azure-storage] Upload complete!
[11:56:26] [deploy-azure-storage] Access your files at: https://spfxwpstorage.blob.core.windows.net/helloworld-webpart
[11:56:26] Finished subtask 'deploy-azure-storage' after 1.92 s
[11:56:26] Finished 'deploy-azure-storage' after 1.93 s
[11:56:26] =====[ Finished ]=====
[11:56:27] Project helloworld-webpart version: 0.0.1
[11:56:27] Build tools version: 2.4.0
[11:56:27] Node version: v6.10.0
[11:56:27] Total duration: 5.94 s
```



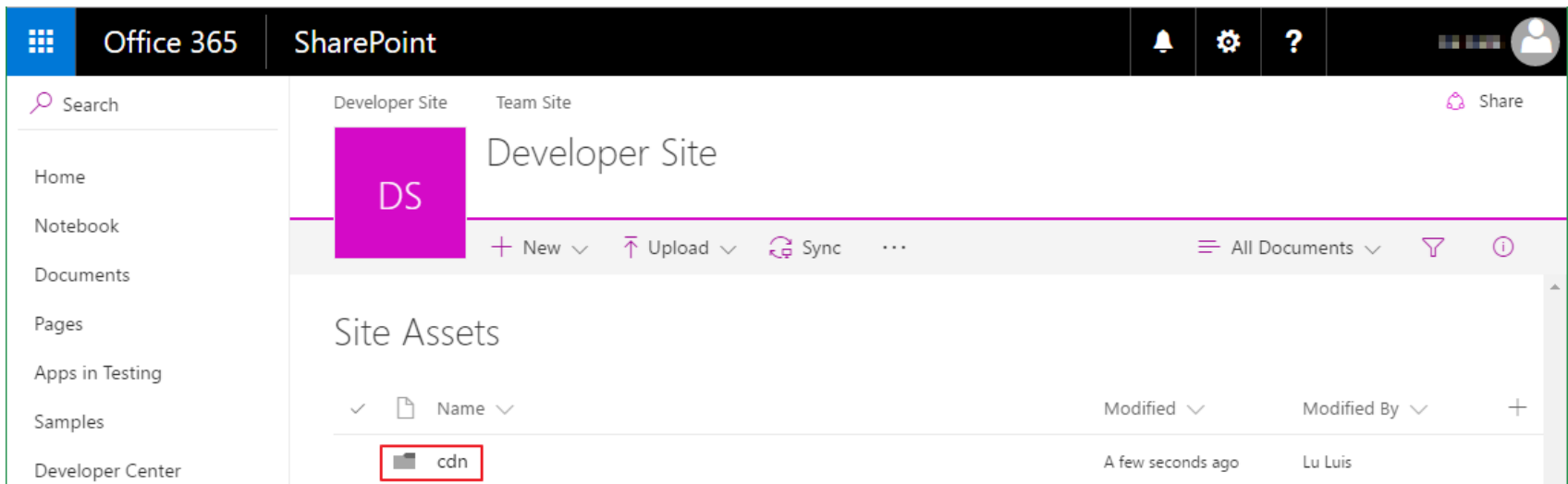
Install the App

- **Go to your Office 365 site**
- **Add the App you just deployed to the SharePoint App Catalog**



Create SharePoint CDN origin

- Go to the Site Assets Document Library
- Create a folder for the CDN origin



Enable the SharePoint CDN origin

- Open the SharePoint Online Management Shell
- Execute the following Powershell commands to enable the CDN, set the CDN origin, and return the CDN Origin ID

```
> $creds = Get-Credential
> Connect-SPOService -Url https://<TENANCY>-admin.sharepoint.com/ -Credential $creds
> Set-SPOTenant -PublicCdnEnabled $true
> Set-SPOTenant -PublicCdnAllowedFileTypes "CSS,EOT,GIF,ICO,JPEG,JPG,JS,MAP,PNG,SVG,TTF,WOFF,TXT"
> New-SPOPublicCdnOrigin -Url "https://<TENANCY>.sharepoint.com/sites/<SITE>/siteassets/cdn"
> Get-SPOPublicCdnOrigins
```

```
PS C:\WINDOWS\System32> Get-SPOPublicCdnOrigins
```

Id	Url
19690057cb318c9369ed0bdea07d8ac5326be	https://[redacted].sharepoint.com/sites/dev/...



Configure the web part to use the CDN

Create CDN base path

`https://publiccdn.sharepointonline.com/<TENANCY>.sharepoint.com/<your-CDN-origin-Id>`

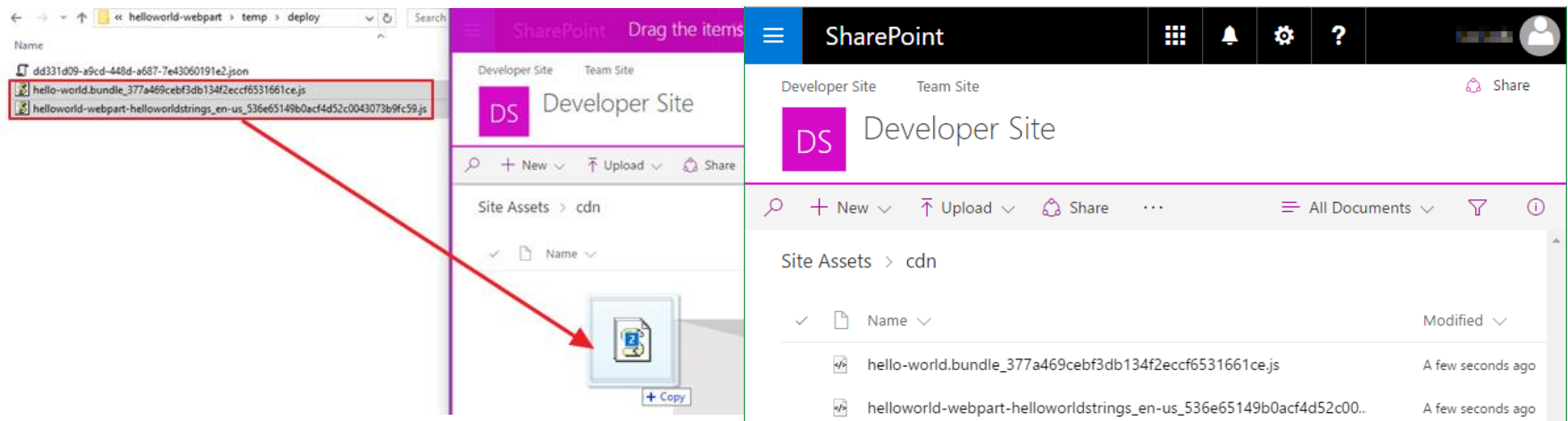
Update the cdnBasePath in the write-manifests.json file

```
{  
  "cdnBasePath": "https://publiccdn.sharepointonline.com/<TENANCY>.sharepoint.com/<your-CDN-origin-Id>"  
}
```



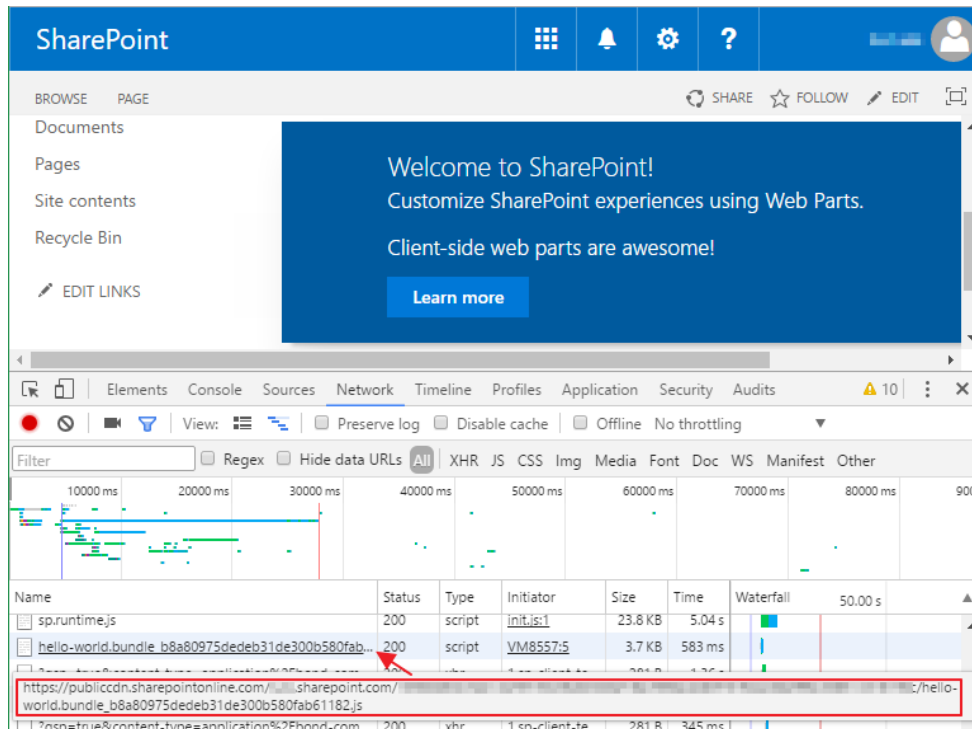
Deploy web part assets to the SharePoint CDN

- Go to the Site Assets Document Library
- Upload the assets for your web part to the CDN enabled folder



Add the web part to a SharePoint page

- After the web part assets are deployed to a CDN location you can preview the web part
- Create page in your SPO site and add the web part to the page
- In the developer console verify the assets are served from the CDN



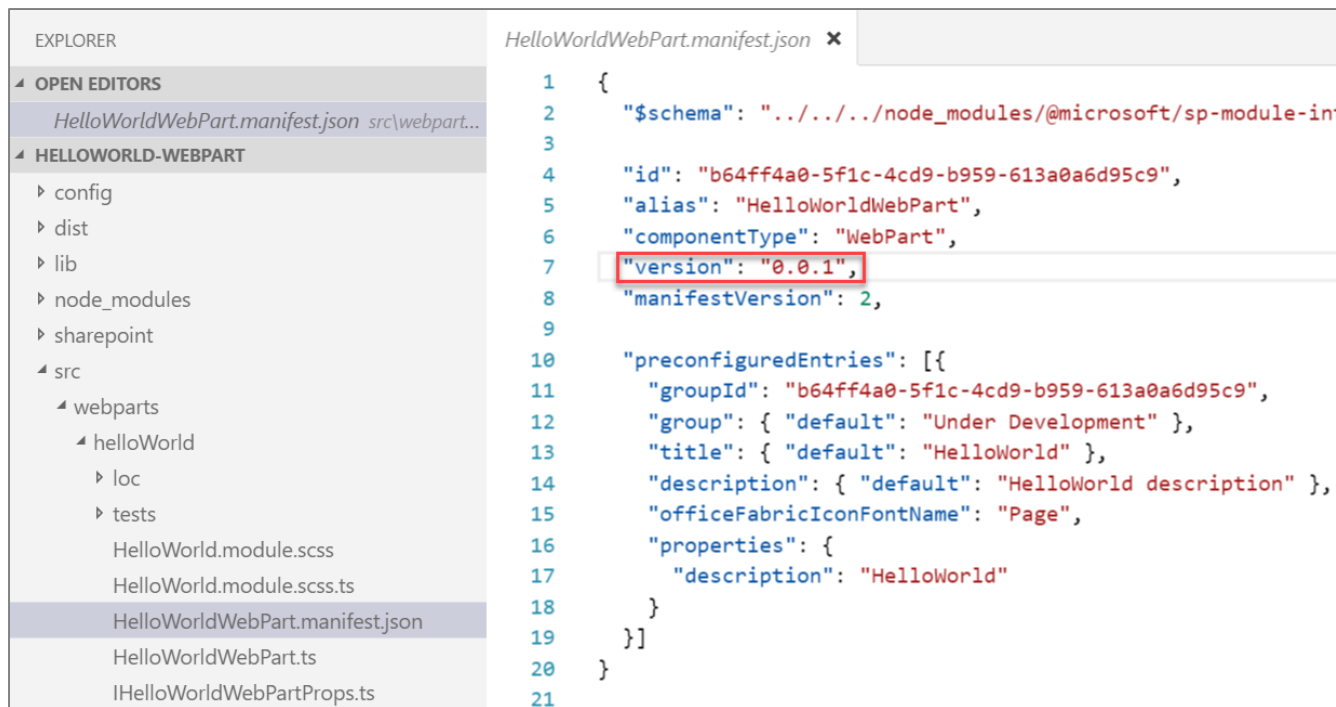
Update Webparts and Increment Version

Update the code or configuration for your web part

Increment the version in the <web part name>.manifest.json file

This sets the version for the web part

Multiple web parts in the same solution may be versioned independently



```
1  {
2    "$schema": "../../node_modules/@microsoft/sp-module-int
3
4    "id": "b64ff4a0-5f1c-4cd9-b959-613a0a6d95c9",
5    "alias": "HelloWorldWebPart",
6    "componentType": "WebPart",
7    "version": "0.0.1",
8    "manifestVersion": 2,
9
10   "preconfiguredEntries": [{
11     "groupId": "b64ff4a0-5f1c-4cd9-b959-613a0a6d95c9",
12     "group": { "default": "Under Development" },
13     "title": { "default": "HelloWorld" },
14     "description": { "default": "HelloWorld description" },
15     "officeFabricIconFontName": "Page",
16     "properties": {
17       "description": "HelloWorld"
18     }
19   }]
20 }
21
```



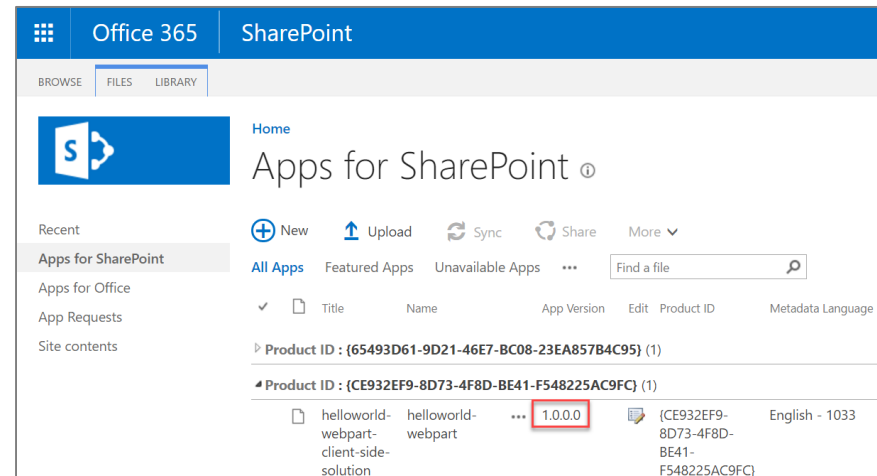
Update Version in package-solution.json

This sets the version for the .sppkg Add-in
This version is displayed in the app catalog



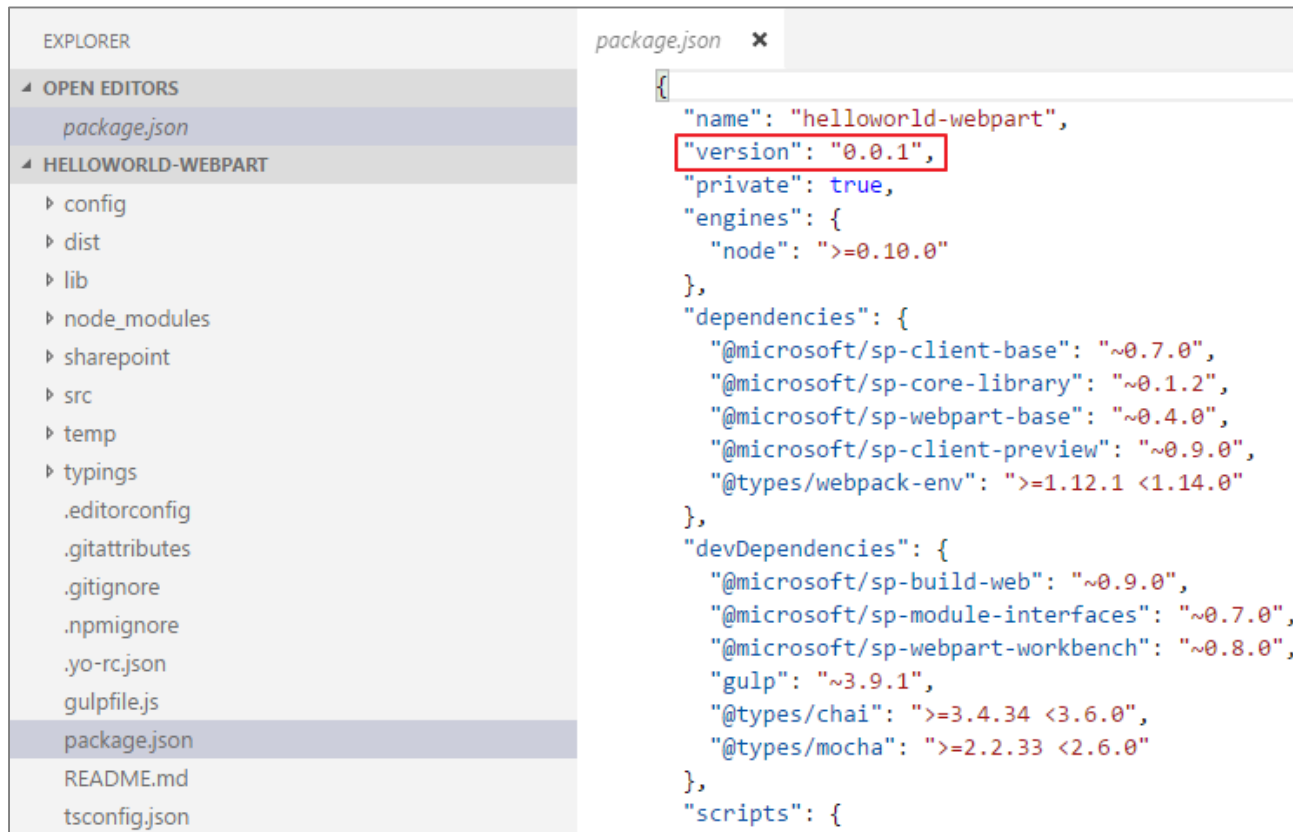
The screenshot shows the VS Code Explorer on the left with the file tree expanded to 'package-solution.json'. The main editor displays the contents of 'package-solution.json', where the 'version' field is highlighted with a red box. The JSON structure is as follows:

```
{
  "solution": {
    "name": "helloworld-webpart-client-side-solution",
    "id": "bd5dbf97-9507-44f6-9e49-a5547f30e5ec",
    "version": "1.0.0.0"
  },
  "paths": {
    "zippedPackage": "solution/helloworld-webpart.sppkg"
  }
}
```



Update the version in the package.json file

- Change the version every time the package changes



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the file structure of a project named 'HELLOWORLD-WEBPART'. The 'package.json' file is selected and highlighted. The main editor area on the right shows the contents of 'package.json'. The 'version' field, which is currently '0.0.1', is highlighted with a red rectangular box. The JSON content is as follows:

```
{
  "name": "helloworld-webpart",
  "version": "0.0.1",
  "private": true,
  "engines": {
    "node": ">=0.10.0"
  },
  "dependencies": {
    "@microsoft/sp-client-base": "~0.7.0",
    "@microsoft/sp-core-library": "~0.1.2",
    "@microsoft/sp-webpart-base": "~0.4.0",
    "@microsoft/sp-client-preview": "~0.9.0",
    "@types/webpack-env": ">=1.12.1 <1.14.0"
  },
  "devDependencies": {
    "@microsoft/sp-build-web": "~0.9.0",
    "@microsoft/sp-module-interfaces": "~0.7.0",
    "@microsoft/sp-webpart-workbench": "~0.8.0",
    "gulp": "~3.9.1",
    "@types/chai": ">=3.4.34 <3.6.0",
    "@types/mocha": ">=2.2.33 <2.6.0"
  },
  "scripts": {
```

