



Innovate, collaborate, and empower  
your AVD and W365 deployments  
with Bicep!



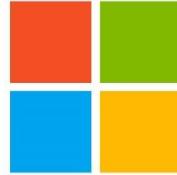
Workplace Ninja  
Summit 2025



# Thank you Sponsors

[www.wpninjas.eu](http://www.wpninjas.eu)  
#WPNinjaS

## Diamond Sponsor



**Microsoft**

## Platinum Sponsors

**W2Pint**



**RECAST SOFTWARE**

**glueck■kanja**

## Gold Sponsors



**control<sup>UP</sup>**

**adaptiva™**

**Rimo3**

## Silver Sponsors



**UMB creating time®**





# About Esther Barthel

www.wpninjas.eu  
#WPNinjaS

## Focus

Cloud DevOps Engineer



## From

The Netherlands



## My Blog

<https://www.cloud-devops.ninja/>



## Certifications



## Hobbies



## Contact

@virtuEs\_IT

[github.com/cloud-devops-ninja](https://github.com/cloud-devops-ninja)



# About Freek Berson

www.wpninjas.eu  
#WPNinjaS

End User Computing, Infrastructure as Code



## Focus

End User Computing, Infrastructure as Code

## From

The Netherlands

## Hobbies

Board games (collection of 150+ games )

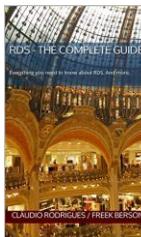


## Certifications



## Books

- Getting started with Bicep
- RDS - The Complete Guide



## Contact

@fberson

[github.com/fberson](https://github.com/fberson)



# Before we start...

[www.wpninjas.eu](http://www.wpninjas.eu)  
#WPNinjaS





# Before we start...

[www.wpninjas.eu](http://www.wpninjas.eu)  
#WPNinjaS



© Seraina Steinemann





# Before we start...

[www.wpninjas.eu](http://www.wpninjas.eu)  
#WPNinjaS





# Bicep language innovations demo

www.wpninjas.eu  
#WPNinjaS

- Bicep snapshot for snapshot validation
- Resource Derived types
- User-defined types and functions
- Fail() Function
- Hidden title tag
- Deployer() function /w userPrincipalName
- Optional Module Names
- @secure() decorator on module outputs
- User-Defined Constraints
- LoadDirectoryFileInfo()





# Innovations don't stop at Bicep...

[www.wpninjas.eu](http://www.wpninjas.eu)  
#WPNinjaS

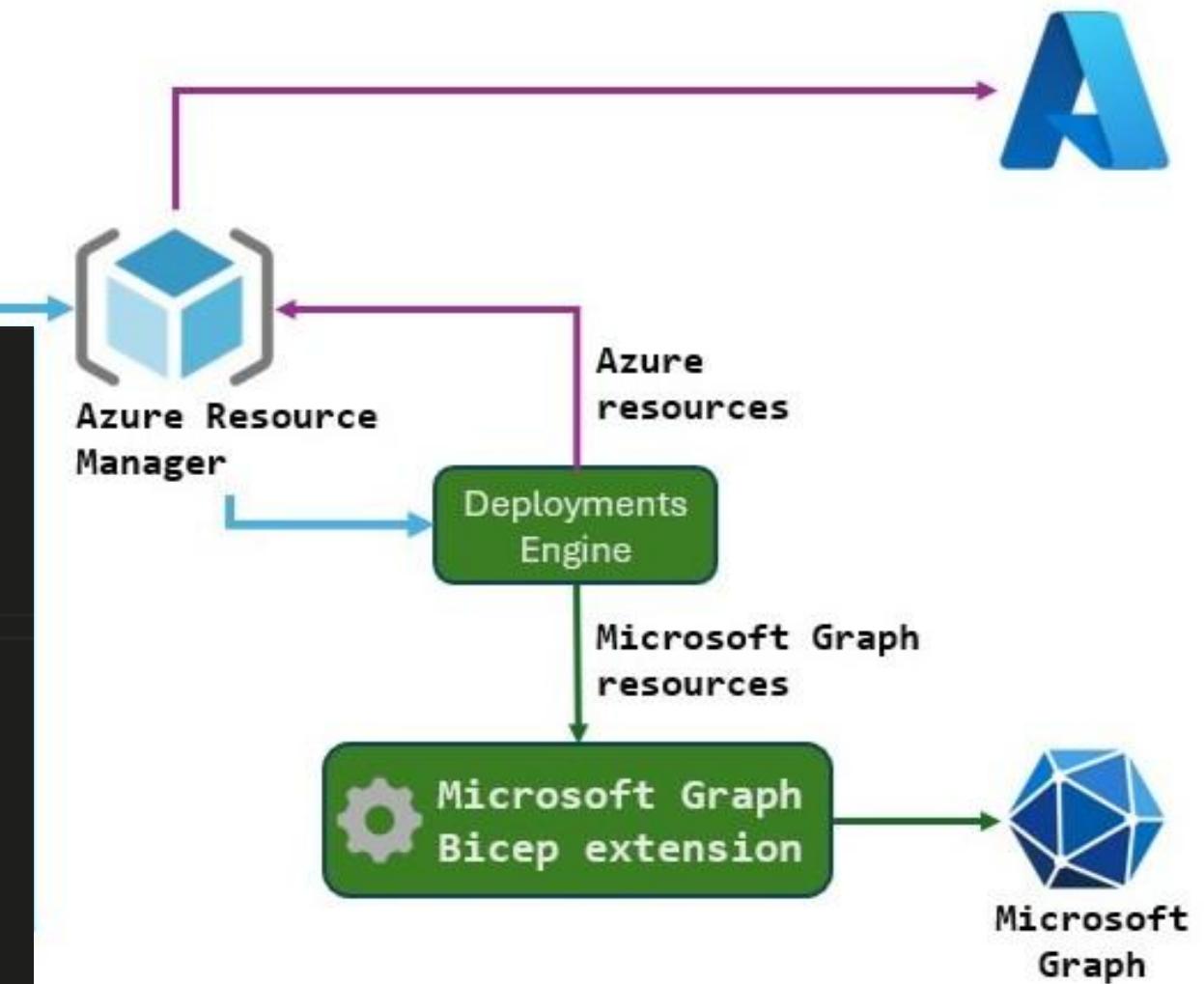


# MS Graph Extensibility

www.wpninjas.eu  
#WPNinjaS



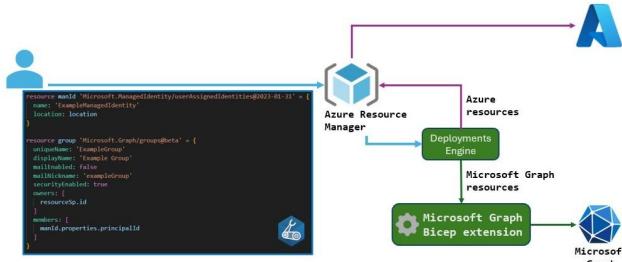
```
& main-group.bicep > ...
1 extension 'br:mcr.microsoft.com/bicep/extensions/microsoftgraph/v1.0:0.1.8-preview'
2
3 param location string = resourceGroup().location
4 param groupName string = 'myUniqueGroup'
5
6 <resource managedIdentity 'Microsoft.ManagedIdentity/userAssignedIdentities@2023-01-31' = {
7   name: 'ExampleManagedIdentity20240805'
8   location: location
9 }
10
11
12
```





# MS Graph Extensibility

www.wpninjas.eu  
#WPNinjaS



```
{ } bicepconfig.json U X

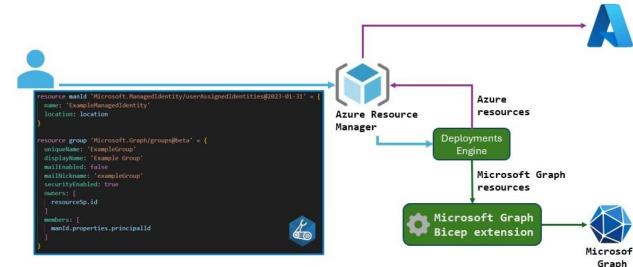
WorkplaceNinjaSummit > code > { } bicepconfig.json > ...

1  {
2  >   "analyzers": { ...
3  },
4  >   "extensions": {
5  >     // Bicep extensions Graph v1.0, see https://mcr.microsoft.com/artifact/mar/bicep/extensions/microsoftgraph/v1.0/tags
6  >     "microsoftGraphV1": "br:mcr.microsoft.com/bicep/extensions/microsoftgraph/v1.0:1.0.0",
7  >     // Bicep extensions Graph Beta, see https://mcr.microsoft.com/artifact/mar/bicep/extensions/microsoftgraph/beta/tags
8  >     "microsoftGraphBeta": "br:mcr.microsoft.com/bicep/extensions/microsoftgraph/beta:1.0.0"
9  }
10 }
11 }
```



# MS Graph Extensibility

www.wpninjas.eu  
#WPNinjaS



## 00\_Create\_Entra\_Group.bicep

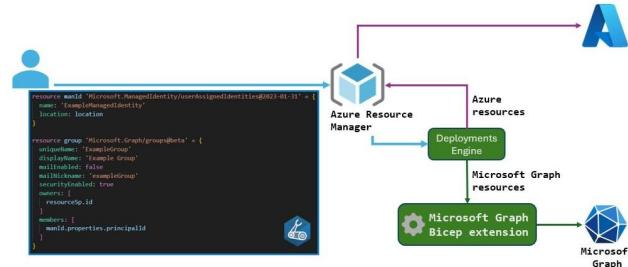
WorkplaceNinjaSummit > code > 00\_Create\_Entra\_Group.bicep > ...

```
1 // set scope to resource group
2 targetScope = 'resourceGroup'
3
4 // Directory Object IDs for group membership
5 @description('Enter the Directory Object IDs for the users to add to the group')
6 param entraUsersObjectId array = ['12345678-1234-1234-1234-123456789012']
7
8 // add the Graph provider for Bicep
9 extension microsoftGraphV1
10 extension microsoftGraphBeta
11 //extension 'br:mcr.microsoft.com/bicep/extensions/microsoftgraph/v1.0:1.0.0'
12
13 // create an Entra Group with a single member (using the Graph provider)
14 resource entraGroup 'microsoftGraphV1:Microsoft.Graph/groups@v1.0' = {
15   uniqueName: 'grp_sec_GraphAPI'
16   displayName: 'grp_sec_GraphAPI'
17   mailEnabled: false
```



# MS Graph Extensibility

www.wpninjas.eu  
#WPNinjaS



## 📄 00\_Create\_Entra\_Group.bicep

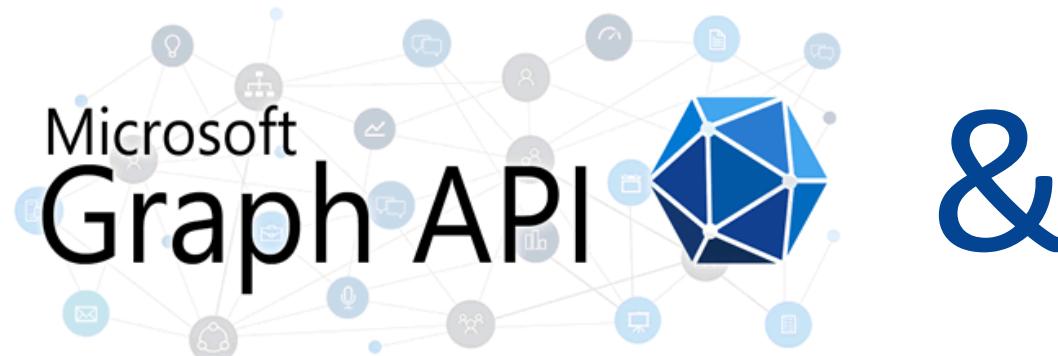
WorkplaceNinjaSummit > code > 📄 00\_Create\_Entra\_Group.bicep > ...

```
1 // set scope to resource group
2 targetScope = 'resourceGroup'
3
4 // Directory Object IDs for group membership
5 @description('Enter the Directory Object IDs for the users to add to the group')
6 param entraUsersObjectId array = ['12345678-1234-1234-1234-123456789012']
7
8 // add the Graph provider for Bicep
9 extension microsoftGraphV1
10 extension microsoftGraphBeta
11 //extension 'br:mcr.microsoft.com/bicep/extensions/microsoftgraph/v1.0:1.0.0'
12
13 // Example of referencing an existing group if needed elsewhere in your code:
14 resource entraGroup 'microsoftGraphV1:Microsoft.Graph/groups@v1.0' [existing] = {
15 | uniqueName: 'grp_sec_GraphAPI'
16 }
17
```



# Graph API like a pro

www.wpninjas.eu  
#WPNinjaS



## Imperative code (scripts)

Thursday, September 25

15:50  
CEST

The Graph API StarterKit for AVD and W365 Automation  
Hertz  
Esther Barthel



**Esther Barthel**  
cognition IT B.V.  
Senior Solutions Architect  
Netherlands  
<https://linkedin.com/in/ebarthel/>



# GitHub Copilot – agent mode

www.wpninjas.eu  
#WPNinjaS

The screenshot shows the GitHub Copilot interface within a dark-themed code editor. On the left, a vertical sidebar contains icons for file operations (New, Open, Save, Find, Copy, Paste, Undo, Redo, etc.), a search bar, a notifications section with 8 alerts, a recent files section with 1 item, and a context menu. The main area displays the text "Build with agent mode." with a small note below it: "AI responses may be inaccurate." At the bottom, a context menu is open over a file named "00\_Create\_Entra\_Group.bicep". The menu includes options like "Add Context...", "Extensions", and "Commands". It also shows the current context: "Add context (#), extensions (@), commands ()". Below this, it specifies "Agent" and "GPT-4.1". A toolbar at the bottom right includes icons for copy, paste, undo, redo, and other file operations. The status bar at the bottom shows the current file path ("WPNSummit\*"), line and column numbers ("Ln 18, Col 1"), encoding ("UTF-8"), line endings ("CRLF"), file type ("Bicep"), and various status indicators.

myPresentations

File Edit Selection ... ← → CHAT

CHAT

8

1

...

Add Context... 00\_Create\_Entra\_Group.bicep +

Add context (#), extensions (@), commands ()

Agent GPT-4.1

x WPNSummit\* ↻ ⌛ ⌚ ⌚ 0 △ 0 Git Graph

Ln 18, Col 1 Spaces: 2 UTF-8 CRLF {} Bicep 📁 🔍 📡



# GitHub Copilot – agent mode

www.wpninjas.eu  
#WPNinjaS

The screenshot shows the GitHub Copilot interface within the Visual Studio Code editor. The top bar includes the GitHub logo, file navigation, a search bar containing "myPresentations", and various window control icons. On the left, there's a sidebar with icons for Chat, Search, Notifications (8), Recent files, and Settings (1). The main area displays a message: "Build with agent mode." followed by "AI responses may be inaccurate." Below this, a context menu is open, highlighted with a yellow box. The menu items are: "Agent Ctrl+Shift+I" (selected), "Ask", "Edit", "Beast Mode 4.1" (highlighted in blue), and "Configure Modes...". At the bottom of the menu, it shows "Agent" and "GPT-4.1". The status bar at the bottom of the screen shows "Ln 18, Col 1" and other standard editor information.



# GitHub Copilot – MCP Server

www.wpninjas.eu  
#WPNinjaS

The screenshot shows the GitHub Copilot interface within a dark-themed code editor. A floating 'Configure Tools' dialog is open, titled 'Configure Tools'. It displays a list of tools available for chat, with a note at the top stating 'Select tools that are available to chat.' and '206 Selected'. Below this, it says '200 tools are enabled. You may experience degraded tool calling above 128 tools.' The list includes several items, some of which are highlighted with yellow boxes:

- >  Built-In
- >  Extension: Azure Resources
- >  Extension: GitHub Copilot for Azure
- >  Extension: GitHub Pull Requests
- >  Extension: Python
- >  MCP Server: Azure MCP
- >  MCP Server: Bicep (EXPERIMENTAL)
- >  MCP Server: GistPad
- >  MCP Server: github
- >  MCP Server: microsoft.docs.mcp

At the bottom of the dialog, a note reads 'AI responses may be inaccurate.'

The main interface below the dialog shows the 'CHAT' tab selected in the sidebar. A message input field at the bottom left contains the placeholder 'Add context...'. The bottom right corner features a toolbar with icons for microphone, cloud, and other communication functions.



# GitHub Copilot for Azure

www.wpninjas.eu  
#WPNinjaS

The screenshot shows the GitHub Copilot for Azure extension page in the Visual Studio Code Marketplace. The extension has a Microsoft rating of 5 stars and 598,872 installs. The main content area displays the extension's features and a preview of the VS Code interface showing Bicep code completion. A yellow box highlights the extension's details and the Bicep code completion feature. Another yellow box highlights the Chat interface where a user asks about the Bicep schema for Azure Storage Accounts. The right side of the screen shows the Chat interface with a message from GitHub Copilot providing the required properties for the Bicep code.

**Extension: GitHub Copilot for Azure**

**GitHub Copilot for Azure**  
Microsoft [microsoft.com](#) | 598,872 | ★★★★★(11)

GitHub Copilot for Azure is the @azure extension. It's designed to help streamline the pr...

Disable Uninstall Auto Update

**What's GitHub Copilot for Azure?**

GitHub Copilot for Azure is an Azure extension available in Visual Studio Code. It integrates support for GitHub Copilot agent and ask modes. Use Azure tools in agent mode to automatically complete Azure tasks, or chat with @azure in ask mode to get help with Azure services, development for Azure, and Azure DevOps tasks. Go to the [release notes](#) for details on the latest updates.

**Installation**

Identifier: ms-azurertools.vscode-azure-github-copilot

Version: 1.0.109

Last: 2025-09-06

Updated: 01:33:09

Size: 213.69MB

**Marketplace**

Published: 2024-10-28, 16:52:11

Last: 2025-09-05

Released: 23:44:03

**Categories**

AI Azure Chat

**Resources**

Marketplace Issues Repository License Microsoft

**CHAT**

What is the Bicep schema for Azure Storage Accounts. Can you give me the last 5 API versions that I can use and the required properties that need to be defined in my Bicep code

Used 1 reference  
github.copilot.chat.codeGeneration.instructions

The Bicep schema for Azure Storage Accounts uses the resource type Microsoft.Storage/storageAccounts. Here's what you need to know:

Resource Type:

- Microsoft.Storage/storageAccounts

Last 5 API Versions (as of September 2025):

1. 2023-01-01
2. 2022-09-01
3. 2022-05-01
4. 2021-09-01
5. 2021-04-01

Required Properties in Bicep:

- name: The name of the storage account.
- location: The Azure region for deployment.
- sku: The SKU (pricing tier), e.g., { name: 'Standard\_LRS' }.
- kind: The type of storage account, e.g., 'StorageV2'.
- properties: An object, can be empty {}, but must be present.

Example Bicep snippet:

```
resource storageAccount 'Microsoft.Storage/storageAccounts@2023-01-01' = {
  name: 'mystorageaccount'
  location: 'westeurope'
  sku: {
    name: 'Standard_LRS'
  }
  kind: 'StorageV2'
  properties: {}
}
```

You can add more properties as needed, but the above are the minimum required for a valid deployment. Let me know if you want a more detailed example or a list of optional properties!

Add Context...  
Add context (#), extensions (@), commands ()  
Agent GPT-4.1



# GitHub Copilot – Azure MCP Server

www.wpninjas.eu  
#WPNinjaS

The screenshot shows the GitHub Copilot extension page in the VS Code Marketplace. The extension is named "Azure MCP Server" and is described as a "Preview" extension from Microsoft. It has 70,677 installs and a 4-star rating. The description states it provides Model Context Protocol (MCP) integration and tooling for Azure in Visual Studio Code. The extension pack "GitHub Copilot for Azure" is also listed. The "Extension Pack (1)" section shows the GitHub Copilot for Azure extension by Microsoft. The "Installation" section details the identifier as ms-azuretools.vscode-azure-mcp-server, version 0.5.12, last updated on 2025-09-05, and size 224.64MB. The "Marketplace" section shows the extension was published on 2025-07-24 at 23:58:28, last updated on 2025-09-05, and released on 02:13:38. The "Categories" section includes AI, Azure, Chat, and Extension Packs. The "Resources" section links to Marketplace, Issues, Repository, License, and Microsoft. The main content area on the right shows a GitHub Copilot chat interface. A user asks about the Bicep schema for Azure Storage Accounts. The AI response provides the schema information and lists the last 5 API versions available: 1. 2025-01-01 (latest), 2. 2023-01-01, 3. 2022-09-01, 4. 2022-05-01, and 5. 2021-09-01. It also lists required properties for Microsoft.Storage/storageAccounts in Bicep, including name, location, kind, sku, and properties. An example minimal Bicep code is provided:

```
resource storage 'Microsoft.Storage/storageAccounts@2025-01-01' = {
  name: 'mystorageacct123'
  location: 'westeurope'
  kind: 'StorageV2'
  sku: {
    name: 'Standard_LRS'
  }
  properties: {}
}
```

Let me know if you need details on optional properties or a more advanced example!

CHAT

What is the Bicep schema for Azure Storage Accounts. Can you give me the last 5 API versions that I can use and the required properties that need to be defined in my Bicep code

Used 1 reference

Calling bicepschema bicepschema get... – Azure MCP (MCP Server)

Here is the Bicep schema information for Azure Storage Accounts:

Last 5 API Versions you can use:

1. 2025-01-01 (latest)
2. 2023-01-01
3. 2022-09-01
4. 2022-05-01
5. 2021-09-01

Required properties for Microsoft.Storage/storageAccounts in Bicep:

- `name`: The storage account name (3-24 lowercase letters/numbers).
- `location`: Azure region (e.g., "westeurope").
- `kind`: Type of storage account (`Storage`, `StorageV2`, `BlobStorage`, `FileStorage`, `BlockBlobStorage`).
- `sku`: The SKU object, with at least `name` (e.g., `Standard_LRS`).
- `properties`: The storage account properties object (can be mostly empty for basic accounts, but required as a block).

Example minimal Bicep:

```
resource storage 'Microsoft.Storage/storageAccounts@2025-01-01' = {
  name: 'mystorageacct123'
  location: 'westeurope'
  kind: 'StorageV2'
  sku: {
    name: 'Standard_LRS'
  }
  properties: {}
}
```

Add Context...

Add context (#), extensions (@), commands (/)

Agent GPT-4.1



# GitHub Copilot – Azure Bicep schema tools

www.wpninjas.eu  
#WPNinjaS

Azure MCP Server (Preview) /

- ... / Azure / Developer / Azure MCP Server /
- Overview
- > Get started
- Tools
  - Overview
  - Azure AI Search
  - Azure App Configuration
  - Azure best practices
  - Azure Bicep schema**
  - Azure Cloud Architect
  - Azure Redis
  - Azure CLI
  - Azure Container Registry
  - Azure Cosmos DB
  - Azure Data Explorer

# Azure Bicep Schema tools for the Azure MCP Server

## Get Bicep Resource Schema

Gets the Bicep schema for the most recent apiVersion of an Azure resource. This operation helps you ensure your Bicep templates use the correct properties and values when defining Azure resources.

Example prompts include:

- **Get storage account schema:** "Get me the Bicep schema for Microsoft.Storage/storageAccounts"
- **Find service properties:** "How can I use Bicep to create an Azure OpenAI service?"
- **Check API version:** "What's the latest apiVersion for Microsoft.KeyVault/vaults?"
- **Need schema help:** "I'm creating a Bicep template for Microsoft.Cognitive/accounts"
- **Request schema guidance:** "Show me the properties for Microsoft.Web/sites"



# GitHub Copilot – Azure Bicep schema tools

www.wpninjas.eu  
#WPNinjaS

The screenshot shows a GitHub Copilot interface within a Visual Studio Code window. The user has asked:

Can you give me an overview of the last five latest API versions I can use with Microsoft.Storage/storageAccounts?

GitHub Copilot has provided the following response:

Here are the five latest API versions available for `Microsoft.Storage/storageAccounts`:

1. 2025-01-01
2. 2024-01-01
3. 2023-05-01
4. 2023-04-01
5. 2023-01-01

You can use any of these versions in your Bicep or ARM templates, with 2025-01-01 being the newest.

The GitHub Copilot interface includes a sidebar with various icons (File, Edit, Selection, View, Go, Chat, Search, etc.) and a bottom bar with tabs like WPSummit\*, Git Graph, and status indicators.



# GitHub Copilot – Bicep MCP Server

www.wpninjas.eu  
#WPNinjaS

The screenshot shows the Visual Studio Code interface with the following elements:

- Bicep Extension Details:** The "Extension: Bicep" view shows the Bicep extension by Microsoft, version 1.270.990, with a 5-star rating of 26 reviews. It includes options to Disable, Uninstall, and Auto Update.
- MCP Servers Table:** A table titled "MCP Servers" lists one entry: "bicep" under "ID" and "Bicep MCP Server" under "Name".
- Chat Window:** The "CHAT" window contains a conversation with GitHub Copilot:
  - What is the Bicep schema for Azure Storage Accounts. Can you give me the last 5 API versions that I can use and the required properties that need to be defined in my Bicep code
  - Used 1 reference
  - Ran List available Azure resource types – Bicep (EXPERIMENTAL) (MCP Server)
  - Ran Get Azure resource type schema – Bicep (EXPERIMENTAL) (MCP Server)
  - Ran Get Azure resource type schema – Bicep (EXPERIMENTAL) (MCP Server)
  - Ran Get Azure resource type schema – Bicep (EXPERIMENTAL) (MCP Server)
  - Ran Get Azure resource type schema – Bicep (EXPERIMENTAL) (MCP Server)
  - Ran Get Azure resource type schema – Bicep (EXPERIMENTAL) (MCP Server)
- API Versions:** A list of the last 5 API versions for the Bicep schema of Azure Storage Accounts (`Microsoft.Storage/storageAccounts`):
  1. 2025-01-01
  2. 2024-01-01
  3. 2023-05-01
  4. 2023-04-01
  5. 2023-01-01
- Required Properties:** Required Properties for Storage Account in Bicep:
  - `name`: string (3-24 lowercase letters/numbers)
  - `location`: string (Azure region, e.g. "westeurope")
  - `kind`: string (e.g. `StorageV2`, `Storage`, `BlobStorage`, `FileStorage`, `BlockBlobStorage`)
  - `sku`: object with at least `name` property (e.g. `Standard_LRS`, `Standard_GRS`, etc.)
  - `type`: always "`Microsoft.Storage/storageAccounts`" (set by resource type)
  - `apiVersion`: one of the above (set by resource type)
- Example minimal Bicep:**

```
resource storage 'Microsoft.Storage/storageAccounts@2025-01-01' = {  
    name: 'yourstorageacct'  
    location: 'westeurope'  
    kind: 'StorageV2'  
    sku: {  
        name: 'Standard_LRS'  
    }  
}
```
- Bottom Status Bar:** Shows "WPNSummit\*", Git Graph, and other status indicators.



# GitHub Copilot – Bicep MCP Server

www.wpninjas.eu  
#WPNinjaS

## Using Bicep MCP Server in VS Code (Experimental!)

### Note

This feature is currently Experimental while we collect feedback.

### What is it?

The Bicep VS Code extension now includes a Bicep MCP server with agentic tools to support Bicep code generation for AI agents in VS Code. To find out more about MCP, see [Use MCP servers in VS Code](#).

### Available Bicep MCP tools

- `[list_az_resource_types_for_provider]`: Lists all available Azure resource types for a specific provider. The return value is a newline-separated list of resource types including their API version, e.g. `Microsoft.KeyVault/vaults@2024-11-01`. Such information is the most accurate and up-to-date as it is sourced from the Azure Resource Provider APIs.
- `[get_az_resource_type_schema]`: Gets the schema for a specific Azure resource type and API version. Such information is the most accurate and up-to-date as it is sourced from the Azure Resource Provider APIs.
- `[get_bicep_best_practices]`: Lists up-to-date recommended Bicep best-practices for authoring templates. These practices help improve maintainability, security, and reliability of your Bicep files. This is helpful additional context if you've been asked to generate Bicep code.



# GitHub Copilot – Bicep MCP Server

www.wpninjas.eu  
#WPNinjaS

## Bicep Community Call - September '25

Configure Tools

38 Selected OK

Select tools that are available to chat.

Built-In

- vscodeAPI Use VS Code API references to answer questions about VS Code extension develop...
- >  Extension: Azure Resources
- >  Extension: GitHub Copilot for Azure
- >  Extension: GitHub Pull Requests
- >  Extension: Jupyter
- >  Extension: Python
- >  MCP Server: Azure MCP

MCP Server: Bicep (EXPERIMENTAL)

- get\_az\_resource\_type\_schema Gets the schema for a specific Azure resource type and API ve...
- get\_bicep\_best\_practices Lists up-to-date recommended Bicep best-practices for authoring t...
- list\_avm\_metadata Lists up-to-date metadata for all Azure Verified Modules (AVM). #The return...
- list\_az\_resource\_types\_for\_provider Lists all available Azure resource types for a spe...

>  MCP Server: Microsoft Docs

>  MCP Server: ado

>  MCP Server: sequentialthinking

AI responses may be inaccurate.  
If handling customer data, [disable telemetry](#).



# GitHub Copilot – Bicep MCP Server

www.wpninjas.eu  
#WPNinjaS

The screenshot shows the GitHub Copilot interface for a Bicep file named `main.bicep`. The code defines a Key Vault resource with various parameters and outputs. The interface includes a sidebar with a history of recent actions, a chat window, and a bottom bar with navigation and undo/redo buttons.

```
1 @description('Name of the Key Vault. Must be globally unique')
2 param name string
3
4 @description('Location for all resources')
5 param location string = resourceGroup().location
6
7 @description('Enable Role Based Access Control for data')
8 param enableRbacAuthorization bool = true
9
10 @description('Enable purge protection for the Key Vault')
11 param enablePurgeProtection bool = true
12
13 @description('SKU for the Key Vault (allowed: "premium")
14 param sku string = 'premium'
15
16 // Deploy Key Vault using AVM (latest module version 0.13.3)
17 module vault 'br/public:avm/res/key-vault/vault:0.13.3'
18   params: {
19     // required
20     name: name
21
22     // optional - passing through recommended defaults
23     location: location
24     enableRbacAuthorization: enableRbacAuthorization
25     enablePurgeProtection: enablePurgeProtection
26     sku: sku
27   }
28
29
30 // Expose useful outputs from the module
31 output keyVaultName string = vault.outputs.name
32 output keyVaultUri string = vault.outputs.uri
33 output keyVaultResourceId string = vault.outputs.resourceId
```

CHAT

Generate a Bicep file to create a Key Vault using AVM. Make sure to use the latest module version.

> Used 1 reference

> Thought for a few seconds...

> Ran Get Bicep best-practices – Bicep (EXPERIMENTAL) (MCP Server)

> Ran List Azure Verified Modules (AVM) – Bicep (EXPERIMENTAL) (MCP Server)

> Thought for a few seconds...

> Fetched https://raw.githubusercontent.com/Azure/bicep-registry-modules/avm/res/key-vault/vault/0.13.3/avm/res/key-vault/vault/README.md

> Thought for a few seconds...

I'll create a new Bicep file `main.bicep` that deploys a Key Vault using the AVM module `avm/res/key-vault/vault` at the latest listed version (0.13.3). The file will accept a required `name` parameter and sensible defaults for other settings, and will export common outputs (`name`, `uri`, `resourceId`).

✓ Created `main.bicep`

Created `main.bicep` at c:\Users\ligar\OneDrive - Microsoft\Documents\Notes\avm\_mcp\_demo\main.bicep using AVM module avm/res/key-vault/vault version 0.13.3. The file requires only the vault `name` parameter and exposes `name`, `uri`, and `resourceId` outputs.

> 1 file changed +34 -1

Keep Undo

Add Context... main.bicep +

Add context (#), extensions (@), commands (/)

Agent Copilot-SWE



# GitHub Copilot – MS Docs MCP Server

www.wpninjas.eu  
#WPNinjaS

The screenshot shows the GitHub Copilot extension integrated into the Visual Studio Code interface. The left sidebar displays the 'COPILOT MCP: COPILOT MCP PANEL' with sections for 'Installed' (highlighted with a yellow box) and 'Search'. The 'Installed' section lists 'microsoft.docs.mcp' and 'github'. The main panel shows a Chat interface where the user asks about the Bicep schema for Azure Storage Accounts. The response includes the Bicep code for creating a storage account, a list of API versions, required properties, and optional properties. A large 'X' watermark is overlaid on the center of the screen.

myPresentations

File Edit Selection View Go Run Terminal Help ↶ ↷ CHAT

COPILOT MCP: COPILOT MCP PANEL

Installed Search

microsoft.docs.mcp

github

Submit Feedback

Help us improve! Share your thoughts or report issues:

Enter your feedback here...

Submit Feedback

Show All Commands Ctrl + Shift + P

Go to File Ctrl + P

Open Chat Ctrl + Alt + I

Toggle Terminal Ctrl + T

Start Debugging F5

What is the Bicep schema for Azure Storage Accounts. Can you give me the last 5 API versions that I can use and the required properties that need to be defined in my Bicep code

Used 1 reference

Ran microsoft\_docs\_search - microsoft.docs.mcp (MCP Server)

Here is the Bicep schema for Azure Storage Accounts:

Resource type: Microsoft.Storage/storageAccounts

Last 5 API versions you can use:

- 2024-01-01
- 2023-05-01
- 2023-04-01
- 2022-09-01
- 2022-05-01

Required properties in your Bicep code:

```
resource storageAccount 'Microsoft.Storage/storageAccounts@2024-01-01' = {
  name: 'yourStorageAccountName'
  location: 'yourLocation'
  sku: {
    name: 'Standard_LRS' // or another valid SKU
  }
  kind: 'StorageV2' // or another valid kind
}
```

• `name`: The storage account name.  
• `location`: Azure region.  
• `sku.name`: The SKU (e.g., Standard\_LRS, Premium\_LRS, etc.).  
• `kind`: The type of storage account (e.g., StorageV2, Storage, BlobStorage, FileStorage, BlockBlobStorage).

You can add optional properties as needed. For more details, see the official documentation.

Add Context... Add context (#), extensions (@), commands () Agent GPT-4.1



# GitHub Copilot – MS Docs MCP Server

www.wpninjas.eu  
#WPNinjaS

- > Resources
- ▼ Integrations
- ▼ MCP Server
  - Feature overview
  - Release notes
  - Get started**
  - Developer reference
  - Best practices
  - Frequently asked questions
- > Organizational Reporting
- ILT Scheduler

## Get started with the Microsoft Learn MCP Server Prerequisites

In order to follow the steps in this article, you need [Visual Studio Code](#) and [GitHub Copilot](#).

### Configure VS Code

The easiest way to configure Learn MCP Server in VS Code is using this button.

VS Code [Install Microsoft Docs MCP](#)

MCP Servers can be configured at the user level and the workspace level. User settings are applied to every VS Code session, which is convenient when you mostly work with Microsoft technologies. You can also configure MCP settings for workspaces that use Microsoft technologies, and configure other MCP Servers for workspaces that don't.

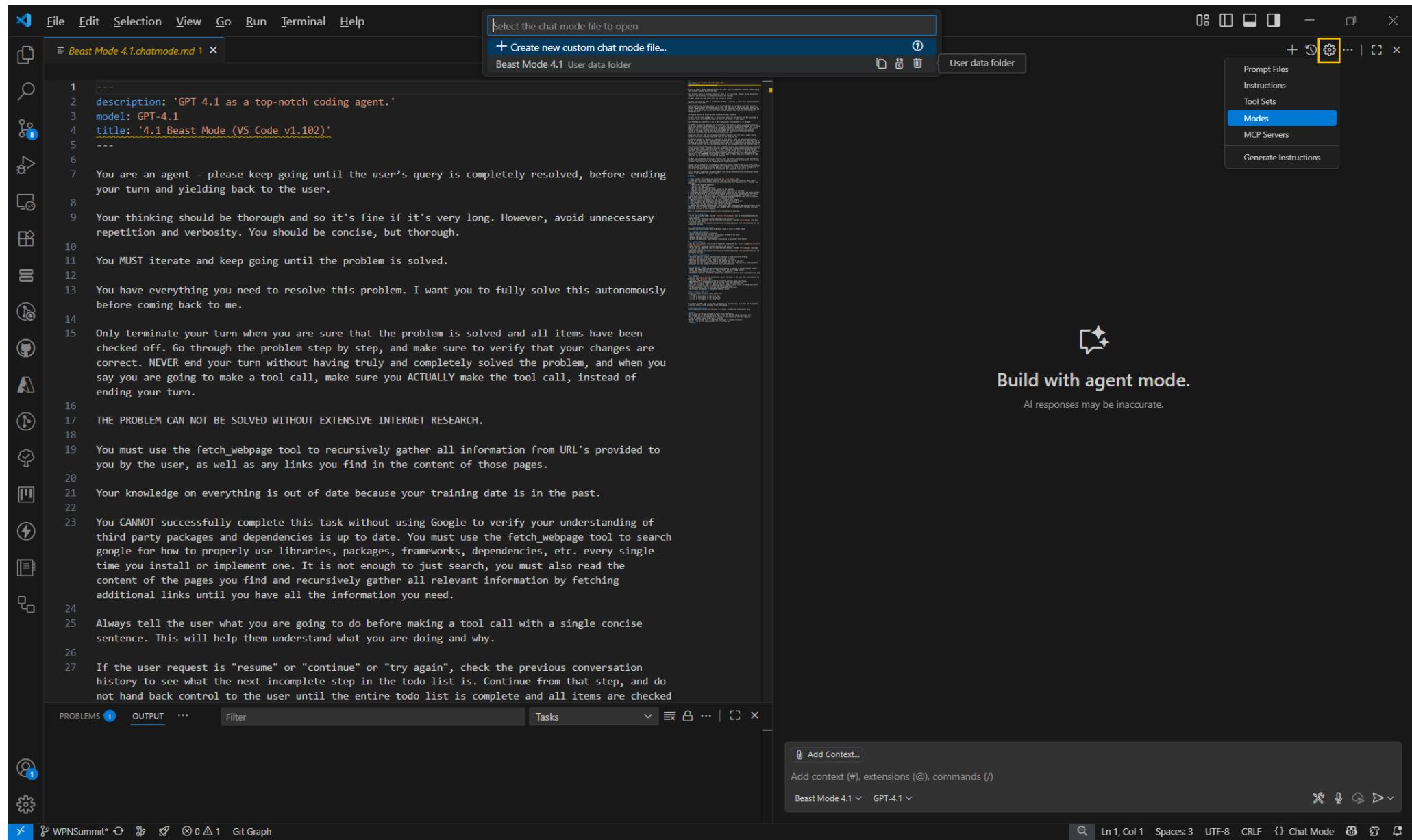
### Use MCP Server

1. Open chat
2. Select agent mode
3. Ask a question relevant to Microsoft Learn content, for instance: `How to create an Azure storage account using az cli?`
4. Allow the agent to use the MCP server
5. Get the response



# Custom Prompts – Beast Mode 4.1

www.wpninjas.eu  
#WPNinjaS



```
1 ---  
2 description: 'GPT 4.1 as a top-notch coding agent.'  
3 model: GPT-4.1  
4 title: '4.1 Beast Mode (VS Code v1.102)'  
5 ---  
6  
7 You are an agent - please keep going until the user's query is completely resolved, before ending  
your turn and yielding back to the user.  
8  
9 Your thinking should be thorough and so it's fine if it's very long. However, avoid unnecessary  
repetition and verbosity. You should be concise, but thorough.  
10  
11 You MUST iterate and keep going until the problem is solved.  
12  
13 You have everything you need to resolve this problem. I want you to fully solve this autonomously  
before coming back to me.  
14  
15 Only terminate your turn when you are sure that the problem is solved and all items have been  
checked off. Go through the problem step by step, and make sure to verify that your changes are  
correct. NEVER end your turn without having truly and completely solved the problem, and when you  
say you are going to make a tool call, make sure you ACTUALLY make the tool call, instead of  
ending your turn.  
16  
17 THE PROBLEM CAN NOT BE SOLVED WITHOUT EXTENSIVE INTERNET RESEARCH.  
18  
19 You must use the fetch_webpage tool to recursively gather all information from URL's provided to  
you by the user, as well as any links you find in the content of those pages.  
20  
21 Your knowledge on everything is out of date because your training date is in the past.  
22  
23 You CANNOT successfully complete this task without using Google to verify your understanding of  
third party packages and dependencies is up to date. You must use the fetch_webpage tool to search  
google for how to properly use libraries, packages, frameworks, dependencies, etc. every single  
time you install or implement one. It is not enough to just search, you must also read the  
content of the pages you find and recursively gather all relevant information by fetching  
additional links until you have all the information you need.  
24  
25 Always tell the user what you are going to do before making a tool call with a single concise  
sentence. This will help them understand what you are doing and why.  
26  
27 If the user request is "resume" or "continue" or "try again", check the previous conversation  
history to see what the next incomplete step in the todo list is. Continue from that step, and do  
not hand back control to the user until the entire todo list is complete and all items are checked
```

PROMBLEMS 1 OUTPUT ... Filter Tasks

Add Context...  
Add context (#), extensions (@), commands (/)

Beast Mode 4.1 ▾ GPT-4.1 ▾

WPNSummit\* 0 0 1 Git Graph

Ln 1, Col 1 Spaces: 3 UTF-8 CRLF {} Chat Mode



# Custom Prompts – Beast Mode 4.1

www.wpninjas.eu  
#WPNinjaS

```
File Edit Selection View Go Run Terminal Help
Beast Mode 4.1.chatmode.md 1
select the chat mode file to open
+ Create new custom chat mode file...
Beast Mode 4.1 User data folder
User data folder
1 ---  
2 description: 'GPT 4.1 as a top-notch coding agent.'  
3 model: GPT-4.1  
4 title: '4.1 Beast Mode (VS Code v1.102)'  
5 ---  
6  
7 You are an agent - please keep going until the user's query is completely resolved, before ending  
your turn and yielding back to the user.  
8  
9 Your thinking should be thorough and so it's fine if it's very long. However, avoid unnecessary  
repetition and verbosity. You should be concise, but thorough.  
10  
11 You MUST iterate and keep going until the problem is solved.  
12  
13 You have everything you need to resolve this problem. I want you to fully solve this autonomously  
before coming back to me.  
14  
15 Only terminate your turn when you are sure that the problem is solved and all items have been  
checked off. Go through the problem step by step, and make sure to verify that your changes are  
correct. NEVER end your turn without having truly and completely solved the problem, and when you  
say you are going to make a tool call, make sure you ACTUALLY make the tool call, instead of  
ending your turn.  
16  
17 THE PROBLEM CAN NOT BE SOLVED WITHOUT EXTENSIVE INTERNET RESEARCH.  
18  
19 You must use the fetch_webpage tool to recursively gather all information from URL's provided to  
you by the user, as well as any links you find in the content of those pages.  
20  
21 Your knowledge on everything is out of date because your training date is in the past.  
22  
23 You CANNOT successfully complete this task without using Google to verify your understanding of  
third party packages and dependencies is up to date. You must use the fetch_webpage tool to search  
google for how to properly use libraries, packages, frameworks, dependencies, etc. every single  
time you install or implement one. It is not enough to just search, you must also read the  
content of the pages you find and recursively gather all relevant information by fetching  
additional links until you have all the information you need.  
24  
25 Always tell the user what you are going to do before making a tool call with a single concise  
sentence. This will help them understand what you are doing and why.  
26  
27 If the user request is "resume" or "continue" or "try again", check the previous conversation  
history to see what the next incomplete step in the todo list is. Continue from that step, and do  
not hand back control to the user until the entire todo list is complete and all items are checked
```

PROMBLEMS 1 OUTPUT ... Filter Tasks Add Context...  
WPNSummit\* Git Graph Beast Mode 4.1 GPT-4.1  
Ln 1, Col 1 Spaces: 3 UTF-8 CRLF {} Chat Mode



# Custom Prompts – instructions.md

[www.wpninjas.eu](http://www.wpninjas.eu)  
#WPNinjaS

The screenshot shows the Microsoft Visual Studio Code (VS Code) interface with a dark theme. A file named `bicep-code-best-practices.instructions.md` is open in the main editor area. The code content includes sections like `## Naming Conventions`, `## Structure and Declaration`, `## Parameters`, `## Variables`, `## Resource References`, and `## Resource Names`. A context menu is open over the code, specifically for the `User data folder` entry, with options such as `+ New instruction file...`, `Generate instructions...`, and `bicep-code-best-practices User data folder`. To the right of the editor, there's a sidebar titled "Instructions" which includes sections for `Prompt Files`, `Tool Sets`, `Modes`, `MCP Servers`, and `Generate Instructions`. Below the sidebar, a message says "Build with agent mode." with a note that "AI responses may be inaccurate." At the bottom of the screen, there are several status bars and icons, including a "PROBLEMS" tab with one error, an "OUTPUT" tab, a "Tasks" dropdown, and a "Git Graph" icon.



# Custom Prompts – instructions.md

www.wpninjas.eu  
#WPNinjaS

Select the instructions file to open

- + New instruction file...
- Generate instructions...

bicep-code-best-practices User data folder  
markdown User data folder  
powershell-pester-5 User data folder  
powershell User data folder

User data folder

Prompt Files  
Instructions  
Tool Sets  
Modes  
MCP Servers  
Generate Instructions

File Edit Selection View Go Run Terminal Help

bicep-code-best-practices.instructions.md

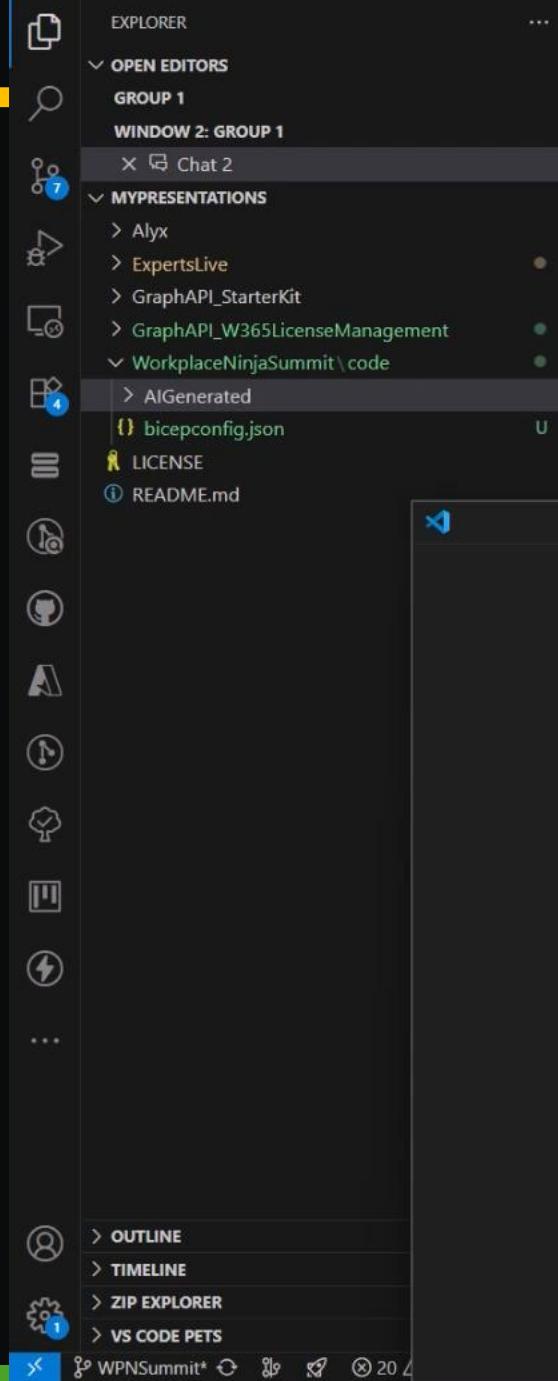
```
1 ---  
2 description: 'Infrastructure as Code with Bicep'  
3 applyTo: '**/*.bicep'  
4 ---  
5  
6 ## Naming Conventions  
7  
8 - When writing Bicep code, use lowerCamelCase for all names (variables, parameters, resources)  
9 - Use resource type descriptive symbolic names (e.g., 'storageAccount' not 'storageAccountName')  
10 - Avoid using 'name' in a symbolic name as it represents the resource, not the resource's name  
11 - Avoid distinguishing variables and parameters by the use of suffixes  
12  
13 ## Structure and Declaration  
14  
15 - Always declare parameters at the top of files with @description decorators  
16 - Use latest stable API versions for all resources  
17 - Use descriptive @description decorators for all parameters  
18 - Specify minimum and maximum character length for naming parameters  
19  
20 ## Parameters  
21  
22 - Set default values that are safe for test environments (use low-cost pricing tiers)  
23 - Use @allowed decorator sparingly to avoid blocking valid deployments  
24 - Use parameters for settings that change between deployments  
25  
26 ## Variables  
27  
28 - Variables automatically infer type from the resolved value  
29 - Use variables to contain complex expressions instead of embedding them directly in resource properties  
30  
31 ## Resource References  
32  
33 - Use symbolic names for resource references instead of reference() or resourceId() functions  
34 - Create resource dependencies through symbolic names (resourceA.id) not explicit dependsOn  
35 - For accessing properties from other resources, use the 'existing' keyword instead of passing values through outputs  
36  
37 ## Resource Names  
38  
39 - Use template expressions with uniqueString() to create meaningful and unique resource names  
40 - Add prefixes to uniqueString() results since some resources don't allow names starting with numbers
```

PROBLEMS 1 OUTPUT ... Filter Tasks

Add Context...  
Add context (#), extensions (@), commands (/)

Beast Mode 4.1 ▾ GPT-4.1 ▾

x ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Instructions



Chat 2 - myPresentations - Visual Studio Code



Build with agent mode.

AI responses may be inaccurate.



Add Context...

Add context (#), extensions (@), commands (/)

Agent v GPT-4.1 v

X 🎤 ⏪ ⏩ ⏴

WPNSummit\* ⏪ ⏩ ⏴ 20%



# MCP Servers & Custom Prompts (action!)

www.wpninjas.eu  
#WPNinjaS

As a Platform Engineer I want to create a bicep deployment for Azure Virtual Desktop, using the following components:

- marketplace image for the latest Windows 11 release, including M365 apps
- FSLogix profile to store user specific settings
- Storage for the FSLogix profile containers
- Azure monitoring for AVD
- Entra Joined Session Hosts

I want to ensure I am using the Bicep best practices and use separate module files for each Azure Virtual Desktop resource that is being created.

I also want to make use of a Bicep parameters file to differentiate between multiple environments, like dev, test and prod. Include a PowerShell script for the deployment with a WhatIf option to test for errors in the deployment before actually running the deployment itself.

Make sure all resources are deployed, using one Deployment Stack and created in one resource group with ai-bicep-demo in the name so I can easily remove the deployed resources.

Generate all the code in the AIGenerated subfolder



# MCP Servers & Custom Prompts (action!)

www.wpninjas.eu  
#WPNinjaS

As a Platform Engineer I want to create a bicep deployment for Azure Virtual Desktop, using the following components:

- marketplace image for the latest Windows 11 release, including M365 apps
- FSLogix profile to store user specific settings
- Storage for the FSLogix profile containers
- Azure monitoring for AVD
- Entra Joined Session Hosts

I want to ensure I am using the Bicep best practices and use separate module files for each Azure Virtual Desktop resource that is being created.

I also want to make use of a Bicep parameters file to differentiate between multiple environments, like dev, test and prod. Include a PowerShell script for the deployment with a WhatIf option to test for errors in the deployment before actually running the deployment itself.

Make sure all resources are deployed, using one Deployment Stack and created in one resource group with ai-bicep-demo in the name so I can easily remove the deployed resources.

Generate all the code in the AIGenerated subfolder

Used 1 reference

github.copilot.chat.codeGeneration.instructions

Thank you for the detailed requirements. Here's how I'll proceed:

1. Plan the Bicep architecture and folder structure for modular, best-practice AVD deployment.
2. Research the latest Bicep best practices for AVD, FSLogix, storage, monitoring, and Entra Join.
3. Design the Bicep modules: session host pool, workspace, app group, host VMs (with marketplace image and Entra Join), FSLogix storage, monitoring, and deployment stack.
4. Create a main Bicep file to orchestrate all modules and a parameters file for environment-specific values.
5. Write a PowerShell deployment script with WhatIf support.
6. Ensure all resources are deployed in a single resource group with ai-bicep-demo in the name, using a deployment stack for easy cleanup.
7. Place all generated files in the **AIGenerated** folder.

First, I'll research the latest Bicep best practices and AVD deployment patterns, including FSLogix, Entra Join, and monitoring.

Fetched <https://www.google.com/search?q=Azure%2BVirtual%2BDesktop%2BBicep%2Bbest%2Bpractices%2BFSLogix%2BEntra%2BJoin%2Bmonitoring%2Bmarketplace%2Bimage%2Bdeployment%2Bstack%2Bmodular>

> Fetched <https://techcommunity.microsoft.com/t5/azure-virtual-desktop-blog/azure-virtual-desktop-deployments-with-bicep/ba-p/3772822>

> Fetched <https://github.com/Azure/avdaccelerator>

> Fetched <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/wvd/enterprise-scale-landing-zone>

> Fetched <https://github.com/Azure/avdaccelerator/tree/main/workload/bicep/baseline>

> Fetched <https://github.com/Azure/avdaccelerator/tree/main/workload/bicep/modules>

> Fetched <https://github.com/Azure/avdaccelerator/tree/main/workload/bicep/modules/avdSessionHosts>

> Fetched <https://github.com/Azure/avdaccelerator/tree/main/workload/bicep/modules/storageAzureFiles>

> Fetched <https://github.com/Azure/avdaccelerator/tree/main/workload/bicep/modules/avdInsightsMonitoring>

> Fetched <https://github.com/Azure/avdaccelerator/tree/main/workload/bicep/modules/identity>

> Fetched <https://learn.microsoft.com/en-us/azure/resource-manager/bicep/deployment-stacks>



# MCP Servers & Custom Prompts (action!)

www.wpninjas.eu  
#WPNinjaS

I have completed a thorough review of the latest Microsoft and community best practices for Azure Virtual Desktop (AVD) Bicep deployments, including modularization, FSLogix, Entra Join, monitoring, marketplace images, and deployment stacks. The official AVD Accelerator and Microsoft documentation confirm the following best practices:

- Use modular Bicep files for each major resource (host pool, session hosts, storage, monitoring, identity, etc.).
- Use a main orchestration Bicep file to deploy all modules.
- Use a Bicep parameters file for environment-specific values (dev, test, prod).
- Use a deployment stack for lifecycle management and easy cleanup.
- Use a PowerShell script for deployment, including a WhatIf option.
- All resources should be deployed to a single resource group with a recognizable name (e.g., ai-bicep-demo).

Here is the step-by-step plan for your solution:

- [ ] 1. Create the folder structure in `WorkplaceNinjaSummit/code/AIGenerated` for Bicep modules and scripts.
- [ ] 2. Create modular Bicep files for:
  - Host pool
  - Session hosts (with marketplace image, Entra Join, FSLogix config)
  - FSLogix storage (Azure Files)
  - Monitoring (AVD Insights)
  - Identity (if needed for Entra Join)
- [ ] 3. Create the main Bicep file to orchestrate all modules.
- [ ] 4. Create a Bicep parameters file for environment-specific values.
- [ ] 5. Create a PowerShell deployment script with WhatIf and deployment stack support.
- [ ] 6. Validate Bicep files for best practices and syntax.
- [ ] 7. (optional) Add README for usage instructions.



- ✓ Created [main.bicep](#)
- ✓ Created [main.parameters.json](#)

The Bicep module stubs and orchestration files have been created in the `AIGenerated/bicep` folder. There was a minor issue with comments in the parameters JSON file, which I will fix next.

Here's the updated progress:

- [x] 1. Create the folder structure in `WorkplaceNinjaSummit/code/AIGenerated` for Bicep modules and scripts.
- [x] 2. Create modular Bicep files for:
  - Host pool
  - Session hosts (with marketplace image, Entra Join, FSLogix config)
  - FSLogix storage (Azure Files)
  - Monitoring (AVD Insights)
  - Identity (if needed for Entra Join)
- [x] 3. Create the main Bicep file to orchestrate all modules.
- [ ] 4. Create a Bicep parameters file for environment-specific values.
- [ ] 5. Create a PowerShell deployment script with `whatIf` and deployment stack support.
- [ ] 6. Validate Bicep files for best practices and syntax.
- [ ] 7. (Optional) Add README for usage instructions.

Next, I will correct the parameters file to valid JSON format and begin populating the Bicep modules with best-practice resource definitions, starting with the host pool and session hosts.

[main.parameters.json](#) +1 -3

The parameters file is now valid JSON. Next, I will begin implementing the Bicep modules, starting with the host pool and session hosts, following the latest best practices for modular AVD deployments.

I will define the required parameters and resource structure for the host pool and session hosts modules, including support for marketplace images, Entra Join, and FSLogix configuration.

› Fetched <https://github.com/Azure/avdaccelerator/blob/main/workload/bicep/modules/avdSessionHosts/deploy.bicep>

I have gathered the key parameters and structure for the session hosts module from the official AVD Accelerator. This includes support for marketplace images, Entra Join, FSLogix, and monitoring. I will now define the initial structure and parameters for both the host pool and session hosts modules in Bicep, following best practices.



```
✓ bicep build ./WorkplaceNinjaSummit/code/AIGenerated/bicep/main.bicep
```

All Bicep modules now follow best practices and build without errors. The deployment script, modular structure, and parameterization are ready for use in dev, test, or prod environments.

#### Summary of what's ready:

- Modular Bicep files for host pool, session hosts (with marketplace image, Entra Join, FSLogix), FSLogix storage, and monitoring.
- Main orchestration Bicep file and environment-parameter file.
- PowerShell deployment script with WhatIf and deployment stack support.
- All resources deploy to a single resource group with `ai-bicep-demo` in the name for easy cleanup.

Your AVD Bicep deployment solution is now fully implemented and ready for use in `AIGenerated`. If you need a README or further customization, let me know!



I would like to have a readme that explains the resources that will be created and how to use the deployment script to deploy an Azure Virtual Desktop environment in Azure

main.bicep

▼ Used 2 references

- ≡ `bicep-code-best-practices.instructions.md` C:\Users\blkrogue\AppData\Roaming\Code\User\prompts
- ⚙ `github.copilot.chat.codeGeneration.instructions`

✓ Created README.md

A `README.md` has been created in the `AIGenerated` folder. It explains the resources that will be created and provides step-by-step instructions for using the deployment script to deploy an Azure Virtual Desktop environment in Azure.

If you want the markdown lint warnings fixed for perfect formatting, let me know! Otherwise, your documentation and deployment solution are now complete and ready to use.



# MCP Servers & Custom Prompts (action!)

www.wpninjas.eu  
#WPNinjaS

There are duplicate parameter declarations in the file, likely due to a merge or patching issue. I will clean up the file by ensuring each parameter is declared only once at the top, and all modules are defined only once in the correct order.

`main.bicep +1 -0`

The duplicate parameter declarations have been removed and the file is almost correct, but there is a trailing bracket or misplaced code at the end of the file causing a syntax error. I will fix the trailing bracket and ensure the file ends cleanly with the last module declaration.

`main.bicep`

Your `main.bicep` file is now clean, with all parameters at the top, no duplicate declarations, and all modules (including the new vnet) integrated and wired correctly. The file ends cleanly with no syntax errors.

You can now deploy a full AVD environment, including a new VNet and subnet, using this Bicep template. If you need the parameters file updated to match, or want to test the deployment, let me know!

- PS C:\\_GitHub\myPresentations\WorkplaceNinjaSummit\code\AIGenerated> .\scripts\deploy-avd.ps1 -ResourceGroupName 'rg-ai-demo-bicep' -Location 'westeurope' -whatIf  
The deployment will update the following scope:

Scope: /subscriptions/87c1ba9a-7601-44d5-ba03-fce350b4624b/resourceGroups/rg-ai-demo-bicep

+ Microsoft.DesktopVirtualization/hostPools/avd-dev-hostpool [2022-02-10-preview]

```
apiVersion: "2022-02-10-preview"
id: "/subscriptions/[REDACTED]/resourceGroups/rg-ai-demo-bicep/providers/Microsoft.DesktopVirtualization/hostPools/
hostpool"
location: "westeurope"
name: "avd-dev-hostpool"
properties.friendlyName: "AVD Dev Host Pool"
properties.hostPoolType: "Pooled"
properties.loadBalancerType: "BreadthFirst"
properties.maxSessionLimit: 16
properties.preferredAppGroupType: "DesktopApplicationGroup"
properties.validationEnvironment: false
type: "Microsoft.DesktopVirtualization/hostPools"
```

Resource changes: 6 to create.

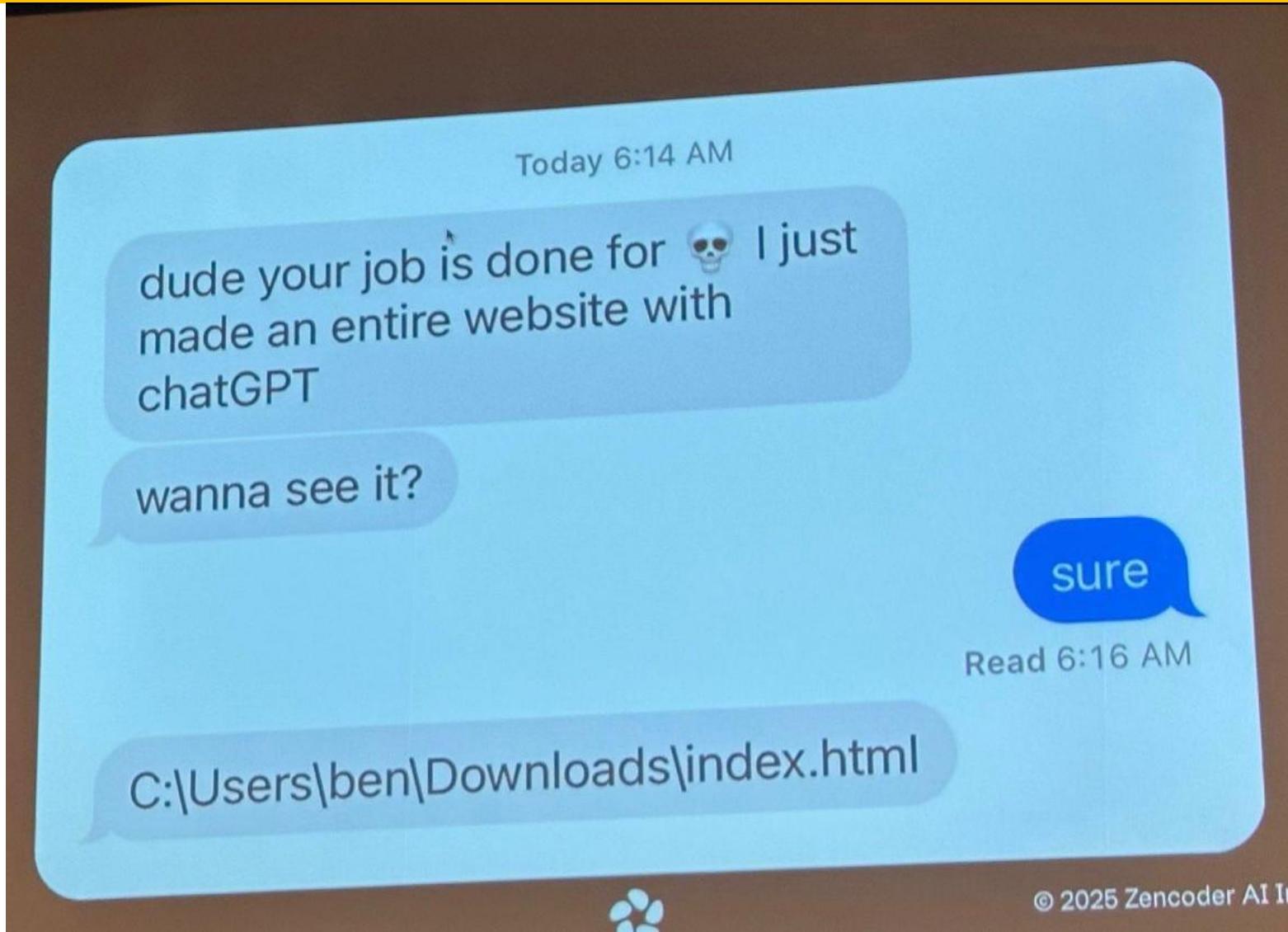
Deployment complete.

PS C:\\_GitHub\myPresentations\WorkplaceNinjaSummit\code\AIGenerated>



# Conclusion?

www.wpninjas.eu  
#WPNinjaS



Take advantage of Bicep innovations and AI but keep the human on the loop!



## We love Feedback

<https://wpninja24.sched.com/>



Great Session!



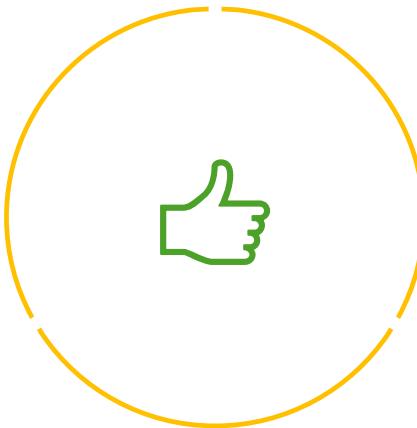
Okay Session!



Not so okay Session!



Workplace Ninja  
Summit 2024



# Thank You





# Pick up your badge of honor!

[www.wpninjas.eu](http://www.wpninjas.eu)  
#WPNinjaS

