Flexing with Bicep 💪

Modern Azure Infrastructure as Code

Garo Yeriazarian
Software Whisperer

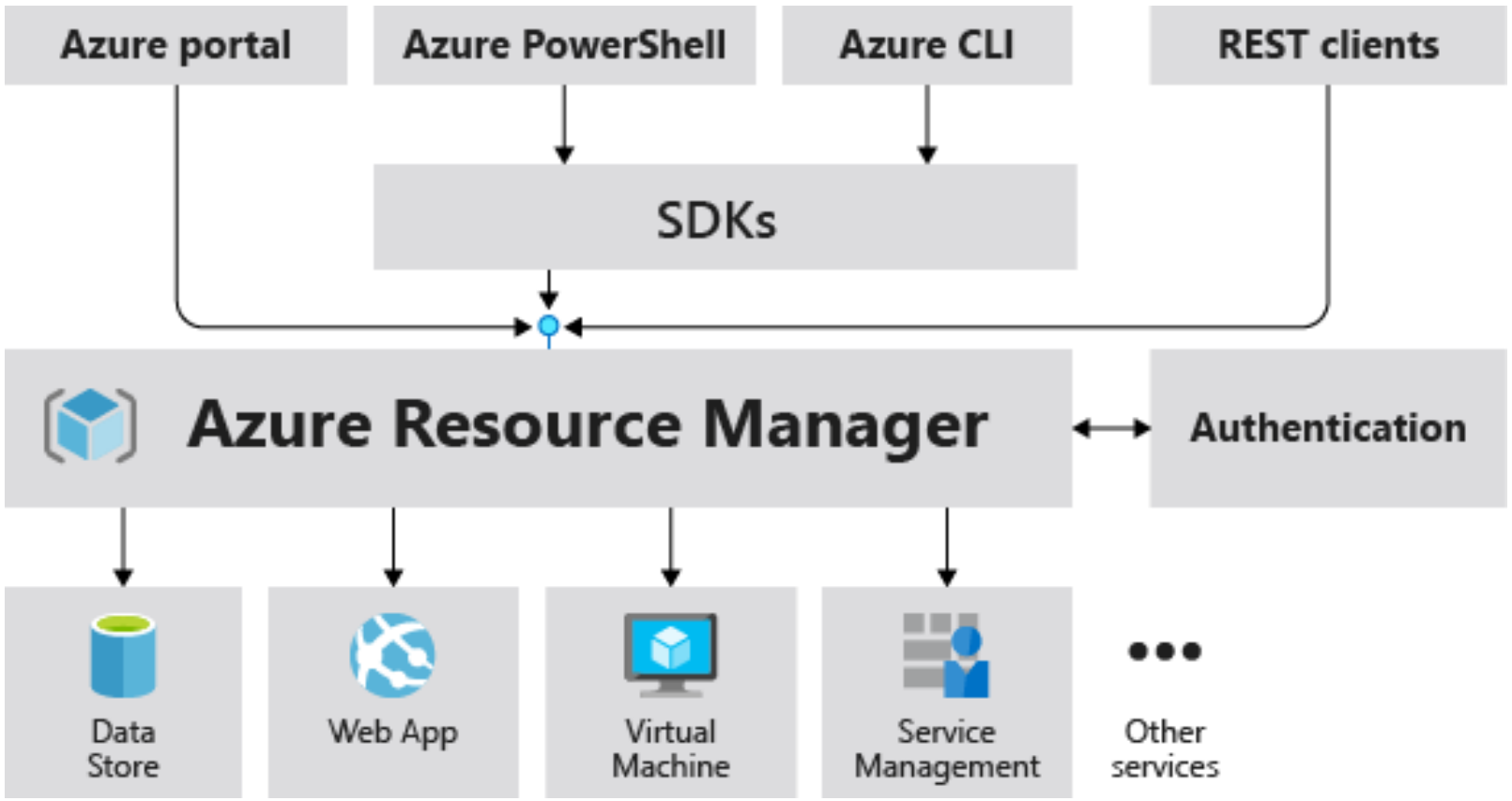
Agenda

What are we going to cover?

- 👀 Overview of Azure Resource Management
- 🦾 What is Bicep and why does it matter?
- 🌎 Infrastructure as Code patterns in the real world
- ➰ Iterative developer workflows

- 🙏 ASK ME QUESTIONS ANYTIME, PLEASE INTERRUPT ME!

Azure Resource Manager (ARM)



Can run outside Azure

Only runs inside Azure

[Azure Resource Manager overview - Azure Resource Manager | Microsoft Learn]

Azure Resource Manager Templates: Why Bicep?

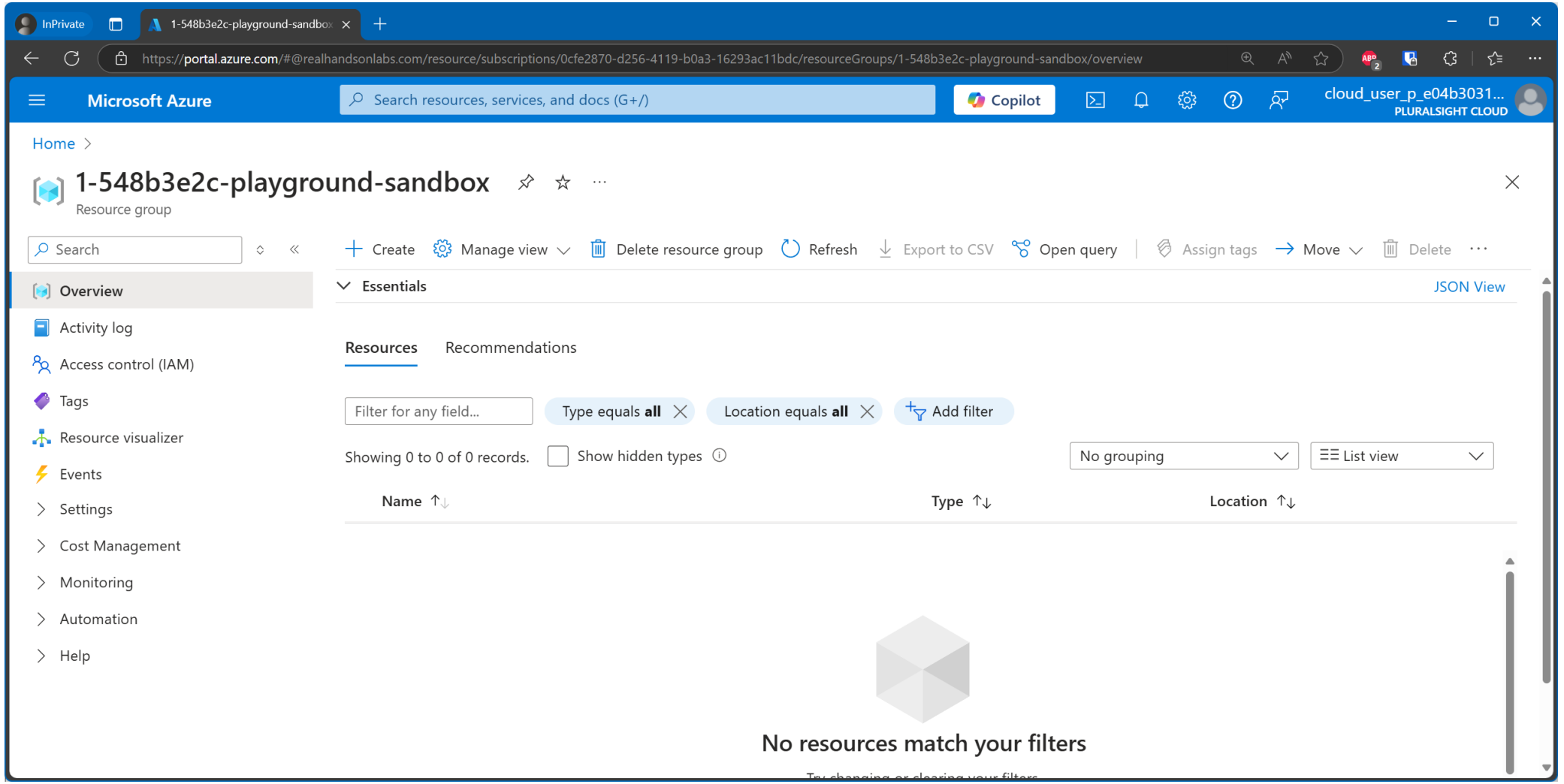
```json
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2021-04-01",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('skuName')]"
      },
      "kind": "StorageV2",
      "properties": {
        "accessTier": "[parameters('accessTier')]"
      }
    }
  ],
  "parameters": {
    "storageAccountName": {
      "type": "string",
      "metadata": {
        "description": "Name of the storage account"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "westus",
      "metadata": {
        "description": "Location for the storage account"
      }
    },
    "skuName": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_ZRS",
        "Premium_LRS"
      ],
      "metadata": {
        "description": "Storage account SKU"
      }
    },
    "accessTier": {
      "type": "string",
      "defaultValue": "Hot",
      "allowedValues": [
        "Hot",
        "Cool"
      ],
      "metadata": {
        "description": "Access tier for the storage account"
      }
    }
  }
}
```

```bicep
@description('Name of the storage account')
param storageAccountName string

@description('Location for the storage account')
param location string = 'westus'

@description('Storage account SKU')
@allowed([
  'Standard_LRS'
  'Standard_GRS'
  'Standard_ZRS'
  'Premium_LRS'
])
param skuName string = 'Standard_LRS'

@description('Access tier for the storage account')
@allowed([
  'Hot'
  'Cool'
])
param accessTier string = 'Hot'

resource storageAccount 'Microsoft.Storage/storageAccounts@2021-04-01' = {
  name: storageAccountName
  location: location
  sku: {
    name: skuName
  }
  kind: 'StorageV2'
  properties: {
    accessTier: accessTier
  }
}
```

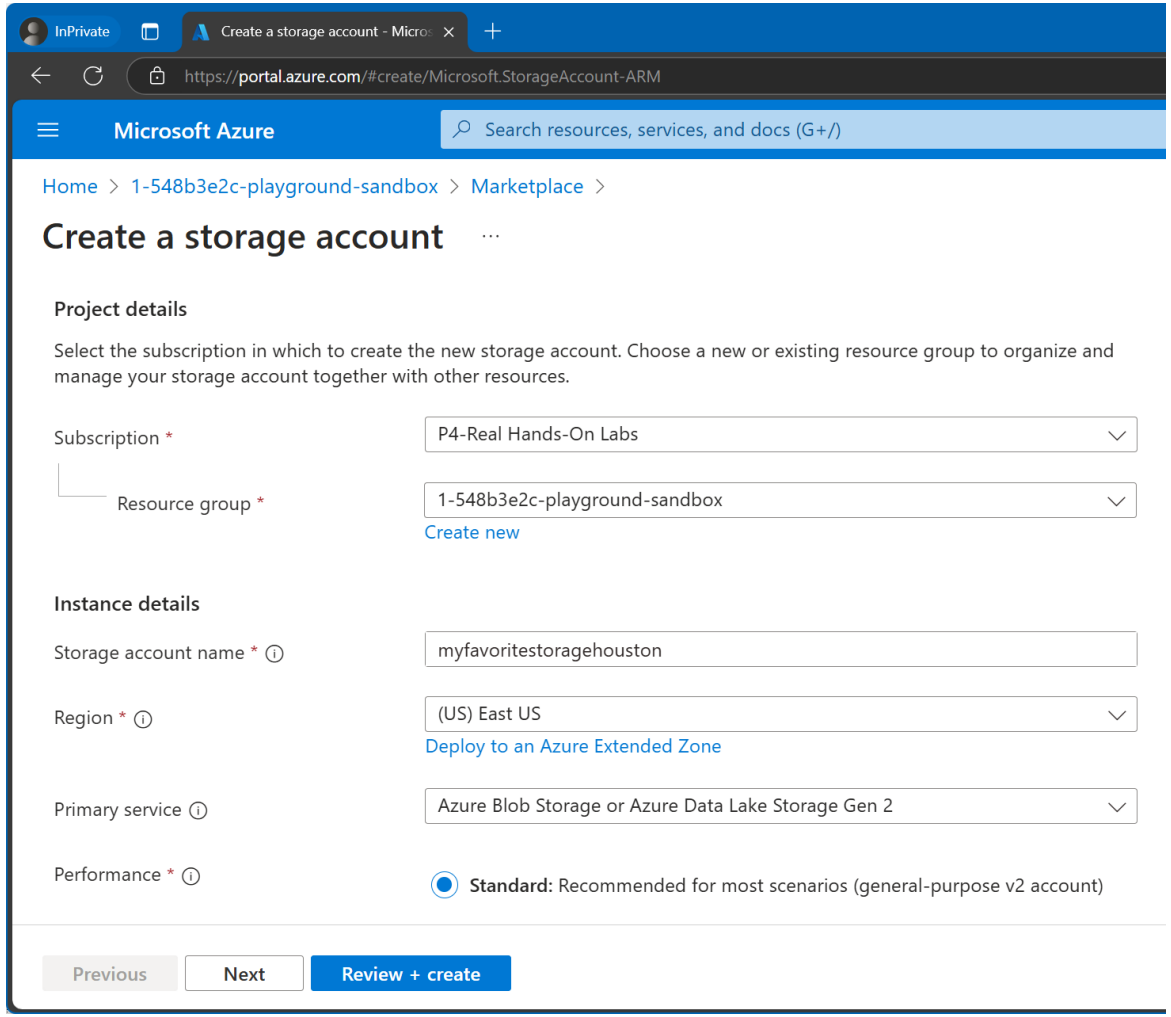
DEMO

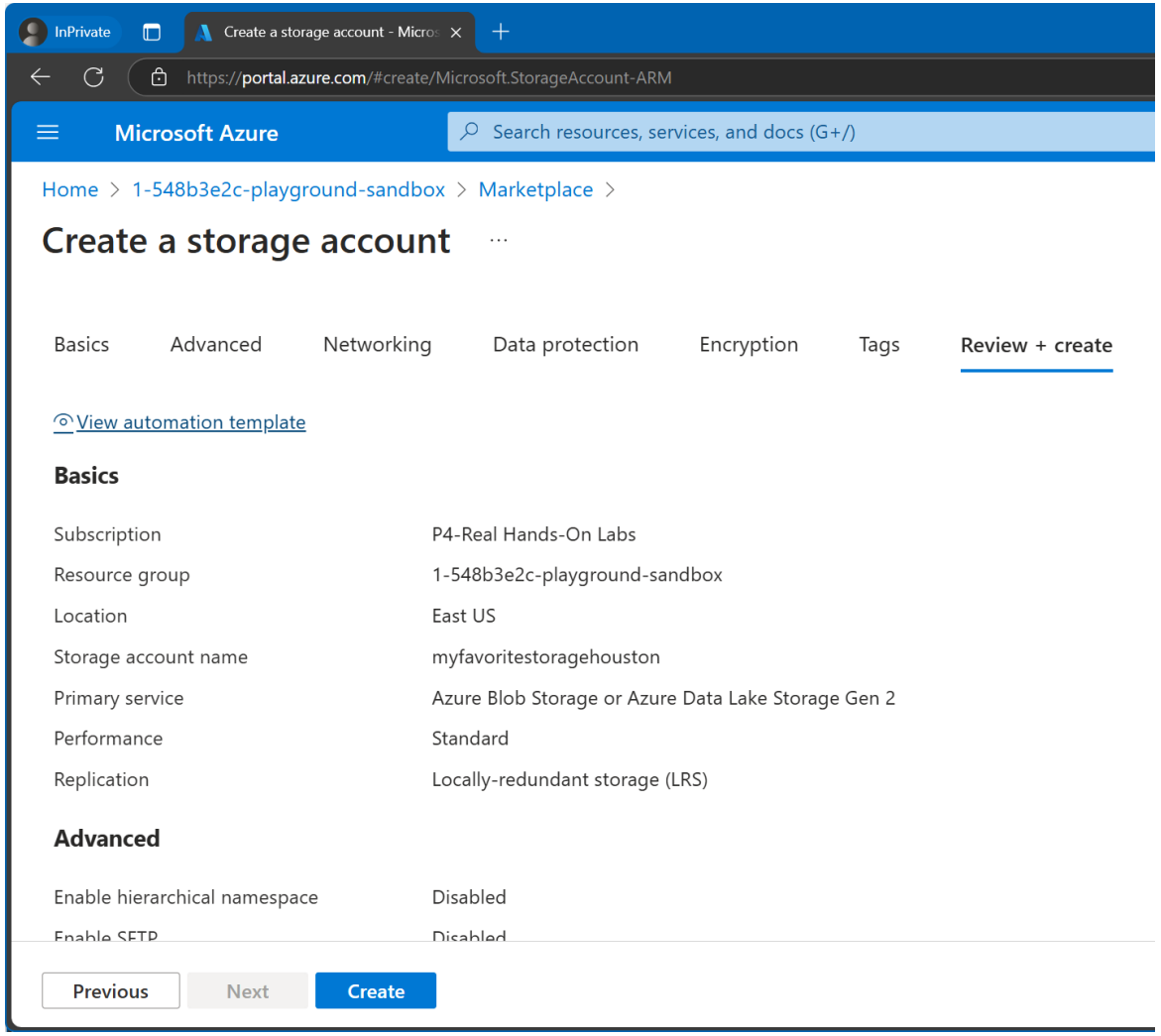
Bicep Templates from Azure Portal

Azure Portal: Initial View

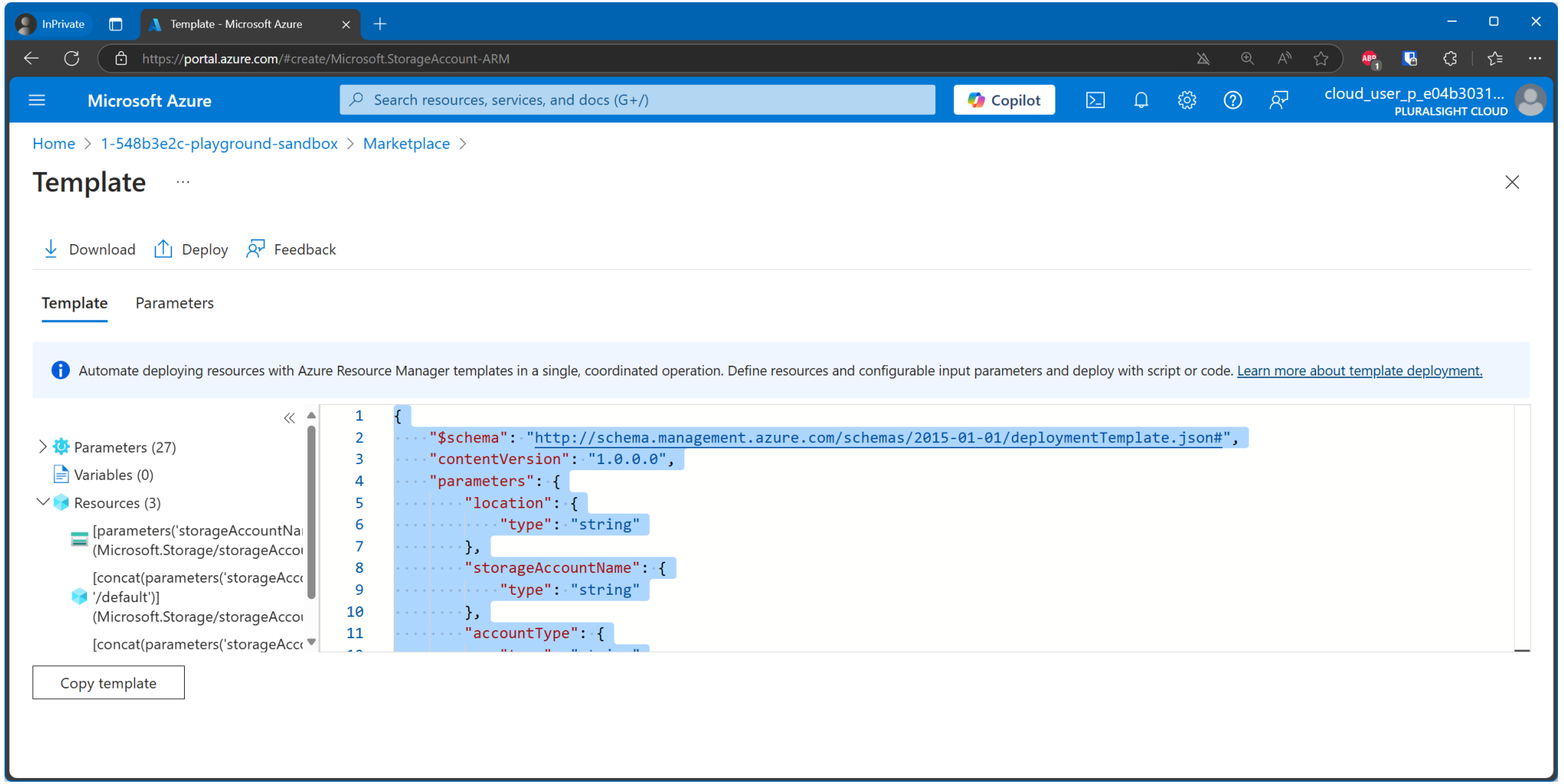


Azure Portal: Create a Storage Account



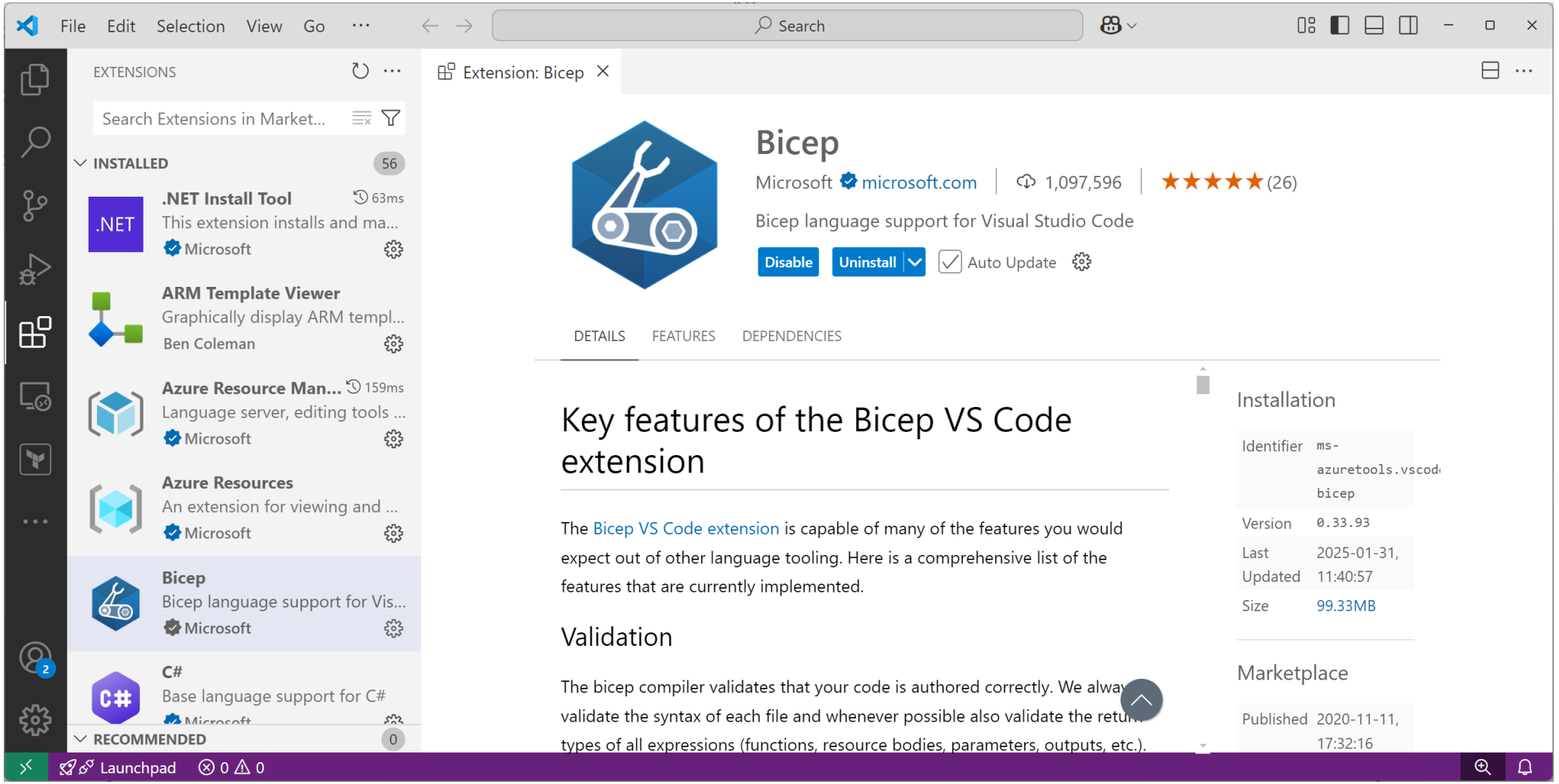


Azure Portal: View Automation Template



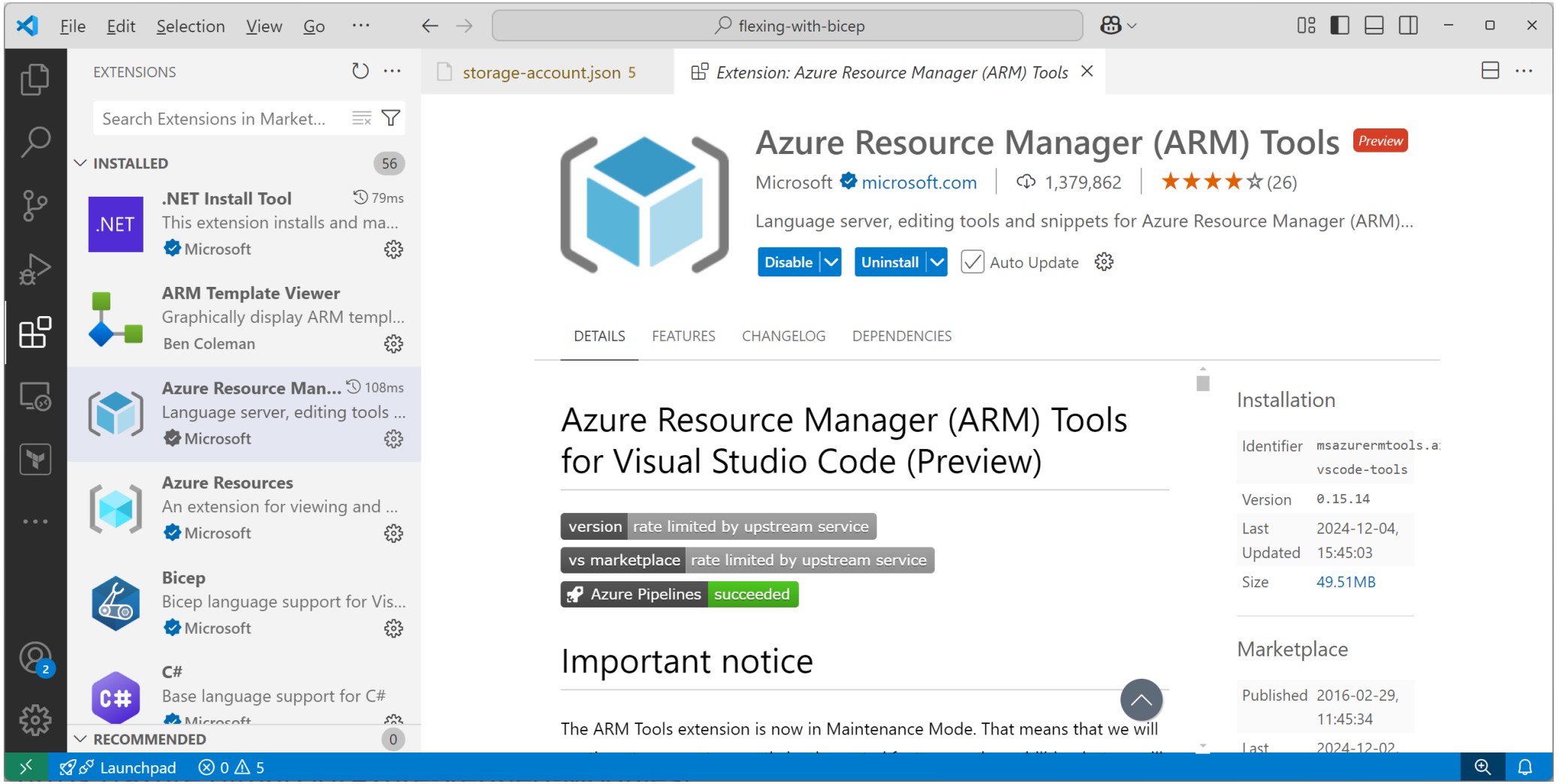
VS Code: Install Bicep Extension

https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-bicep

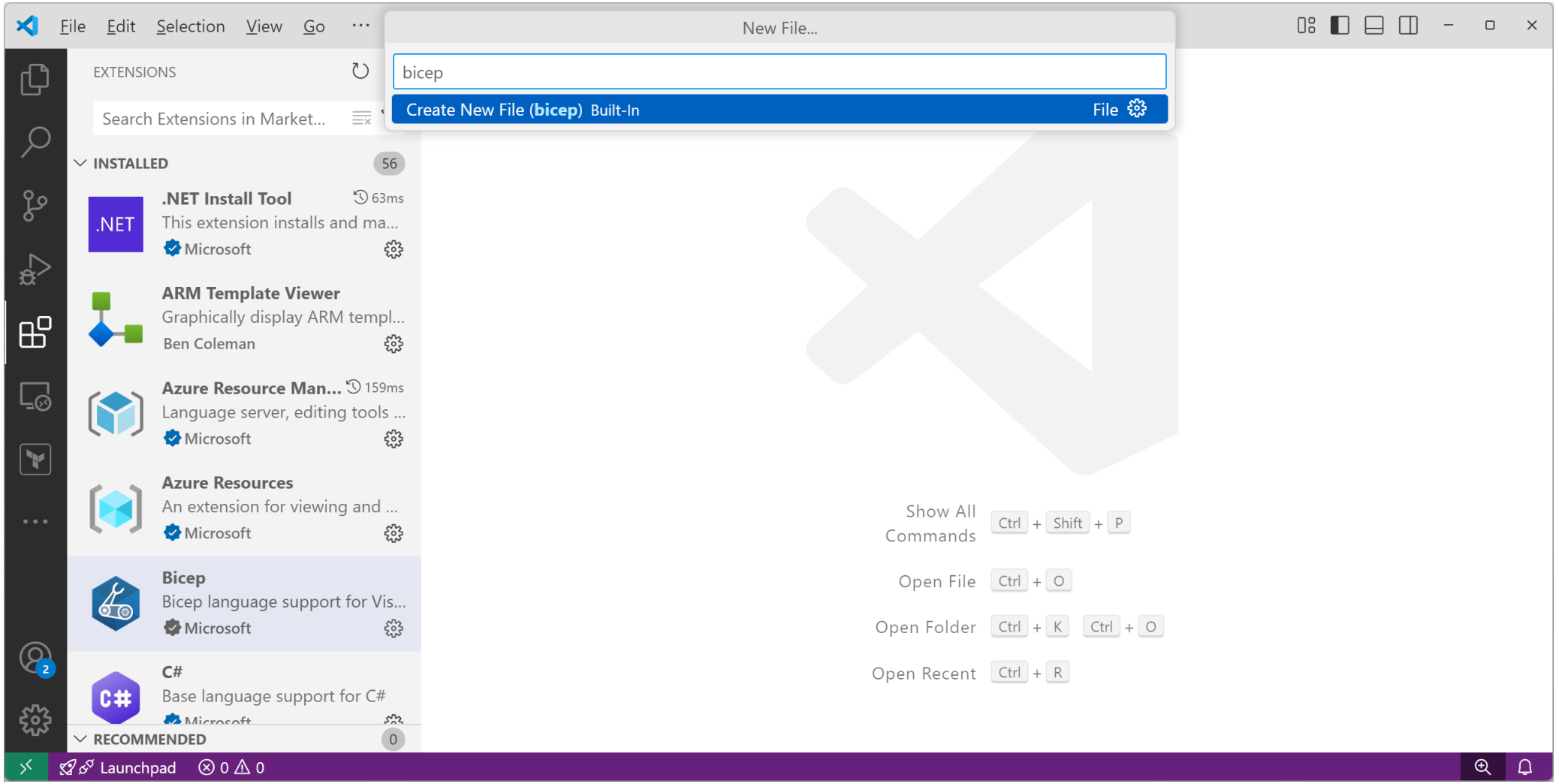


VS Code: Install ARM Template Extension

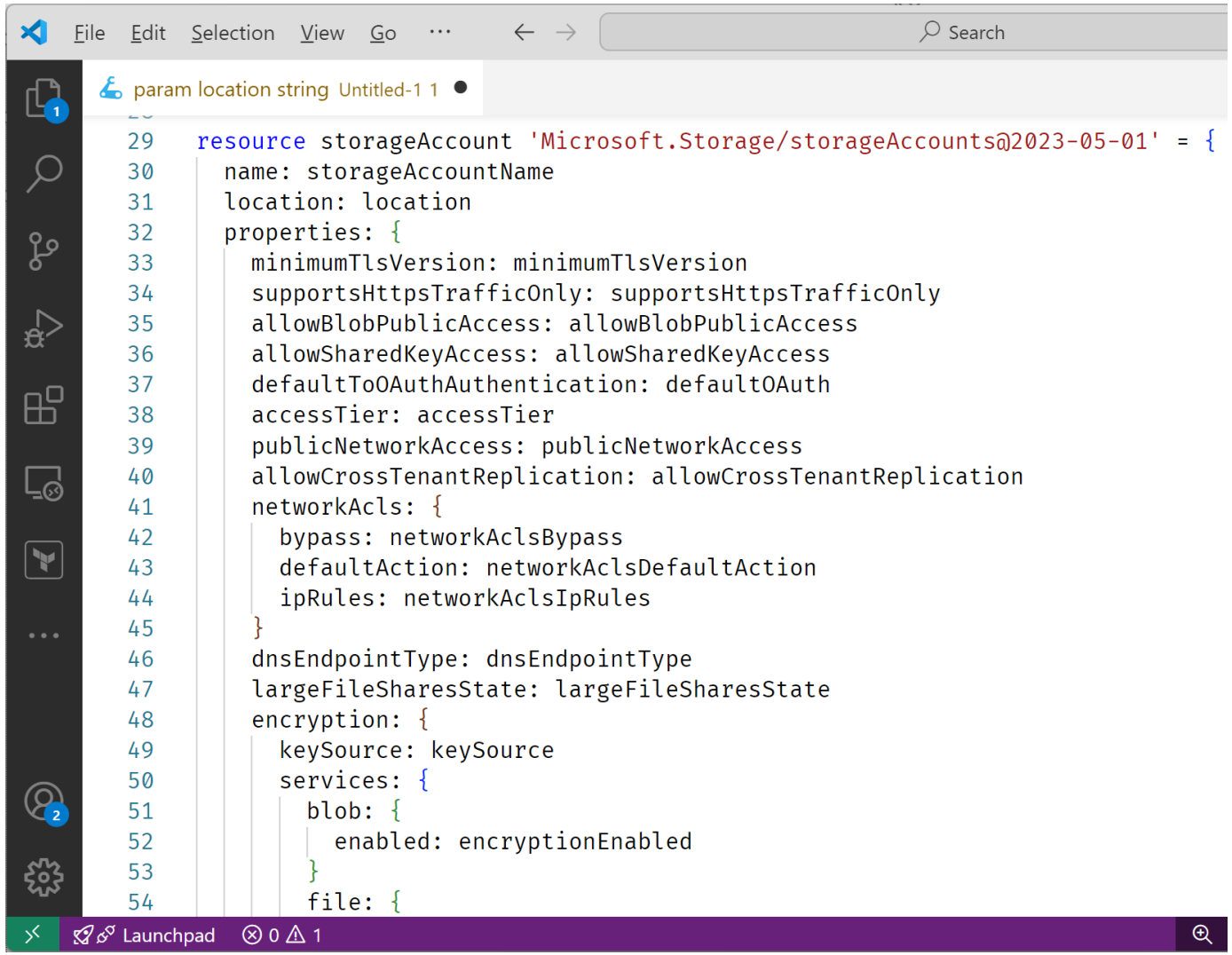
https://marketplace.visualstudio.com/items?itemName=msazurermtools.azurerm-vscode-tools



VS Code: New Bicep File



VS Code: Paste ARM Template JSON



JSON will be decompiled to Bicep! 😻

But this doesn't make life that much easier. 😿

Why Bicep?

Azure Bicep

Objective

- Drastically simplify the authoring experience of Azure resources with a cleaner syntax, improved type safety, and better support for modularity and code re-use.

Significant Work History

- **2021:**
  - Initial release. Simplified syntax for ARM templates.
  - Enhanced error handling and diagnostic capabilities.
- **2022:**
  - Loops, conditionals, and parameterization.
  - Integration with development tools and CI/CD pipelines.
- **2023:**
  - Bicep Public Module Registry launched.
  - User-defined types added.
- **2024:**
  - User-defined functions added.
  - Public introduction of Azure Verified Modules, pre-configured and rigorously tested building blocks.

Strengths

- Not JSON, excellent expressivity and modularity support
- Flexible, expressive language and type system
- Active community inside and outside Microsoft
- Some extra provider support: Entra ID, Kubernetes
- Built into Azure, easy learning curve (now), no state files!
- Rich set of patterns and best practices: AVM

Weaknesses

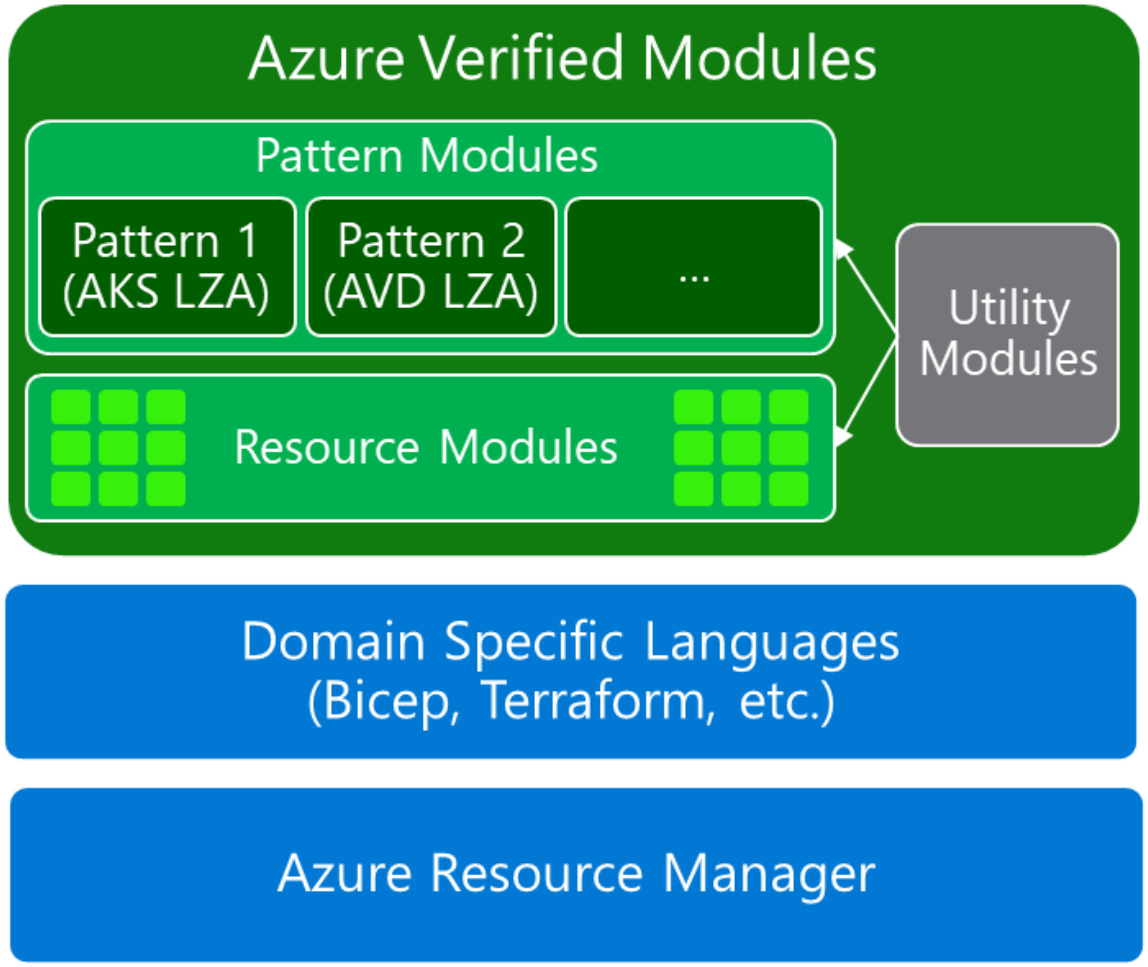
- Azure + Azure Resource Manager support only
- Hard to validate without deploying (what-if support is iffy)
- Naming things is HARD
- Sometimes annoying documentation (but it’s getting better)
- Intellisense for selecting modules from custom registries is not there yet (but accessing the main public registry is not terrible)

Expressive Language Features

- User-defined types are super flexible
  - Bicep uses a type system like TypeScript
  - User-defined types in Bicep - Azure Resource Manager | Microsoft Learn
- Use the import keyword to bring in types and other exported things from outside your Bicep file
  - Imports in Bicep - Azure Resource Manager | Microsoft Learn
- Find yourself doing the same transformations over and over? Look at user-defined functions
  - User-defined functions in Bicep - Azure Resource Manager | Microsoft Learn
- Don’t repeat yourself, use the built-in functions and lambda expressions to process collections
  - Bicep functions - lambda - Azure Resource Manager | Microsoft Learn
  - This comes in handy when doing complex things with Application Gateway (you should try to move to FrontDoor instead), or with Virtual Networks
- Safe dereferencing and null-coalescing:
  - Bicep safe-dereference operator - Azure Resource Manager | Microsoft Learn
  - Bicep logical operators - Azure Resource Manager | Microsoft Learn

Modules and Organization

Using Modules to Organize Infrastructure



Realm of Laziness

😻

Work up here!

Realm of Overwork

😿

Don’t work down here!

https://azure.github.io/Azure-Verified-Modules/

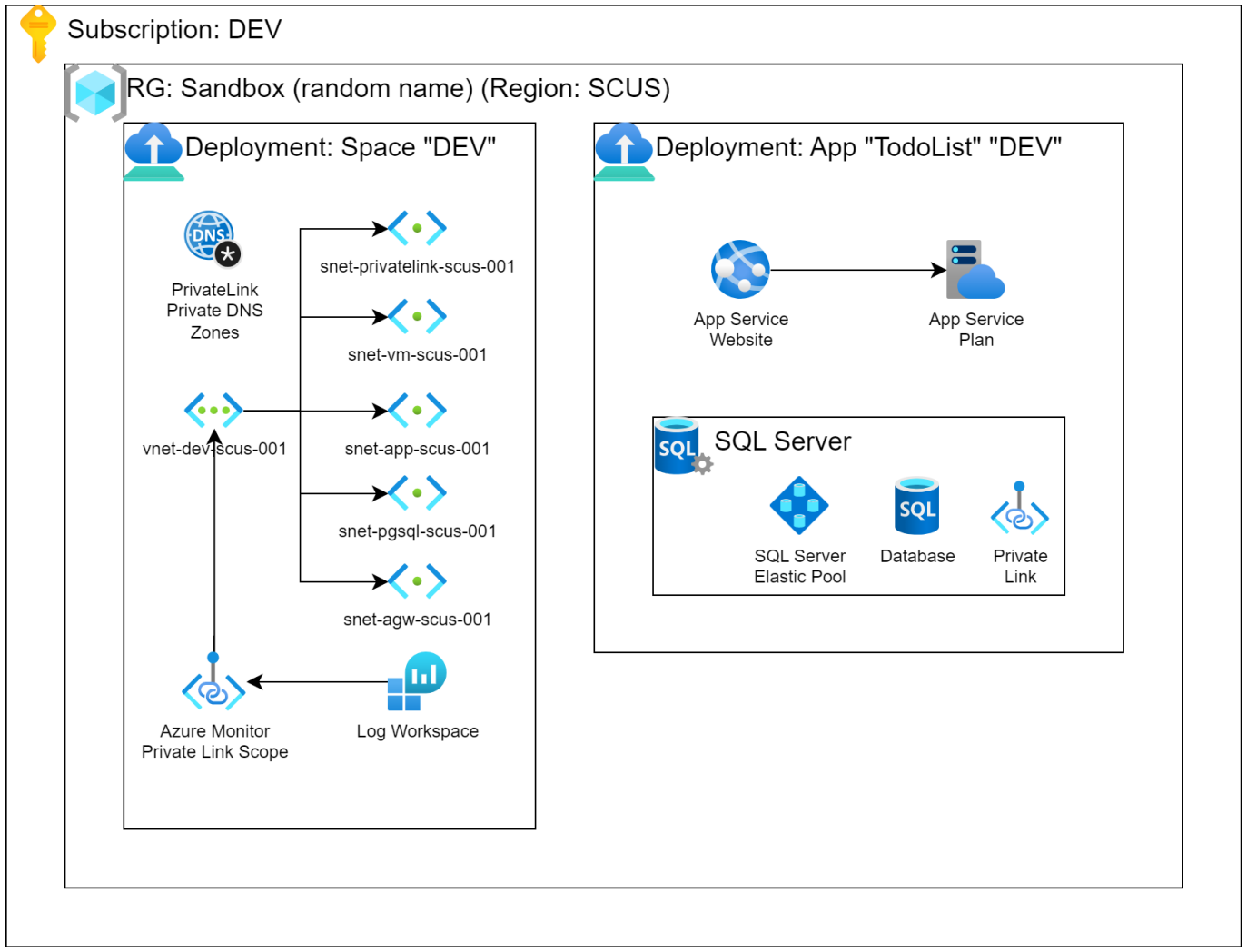
DEMO

Azure Verified Modules

DEMO

Abstracting the Common Application Patterns

Cloud Infrastructure



- Use Deployment Stacks to organize resources into managed blocks
- We are cheating a bit due to ACloudGuru's sandbox limitations and using only one Resource Group
- Everything should be kept on the private network, and we can later add a FrontDoor to allow public access to the App Service
- There are some missing integrations like setting up a connection string on the site to the database, and some Key Vault fun

Modern Infrastructure as Code

Modern Infrastructure as Code

Platform Team

- Establish and **Publish** a repository of **patterns** and **best practices**
- Enable **ephemeral** developer sandboxes for **faster iteration** (Azure Deployment Groups / Stacks)

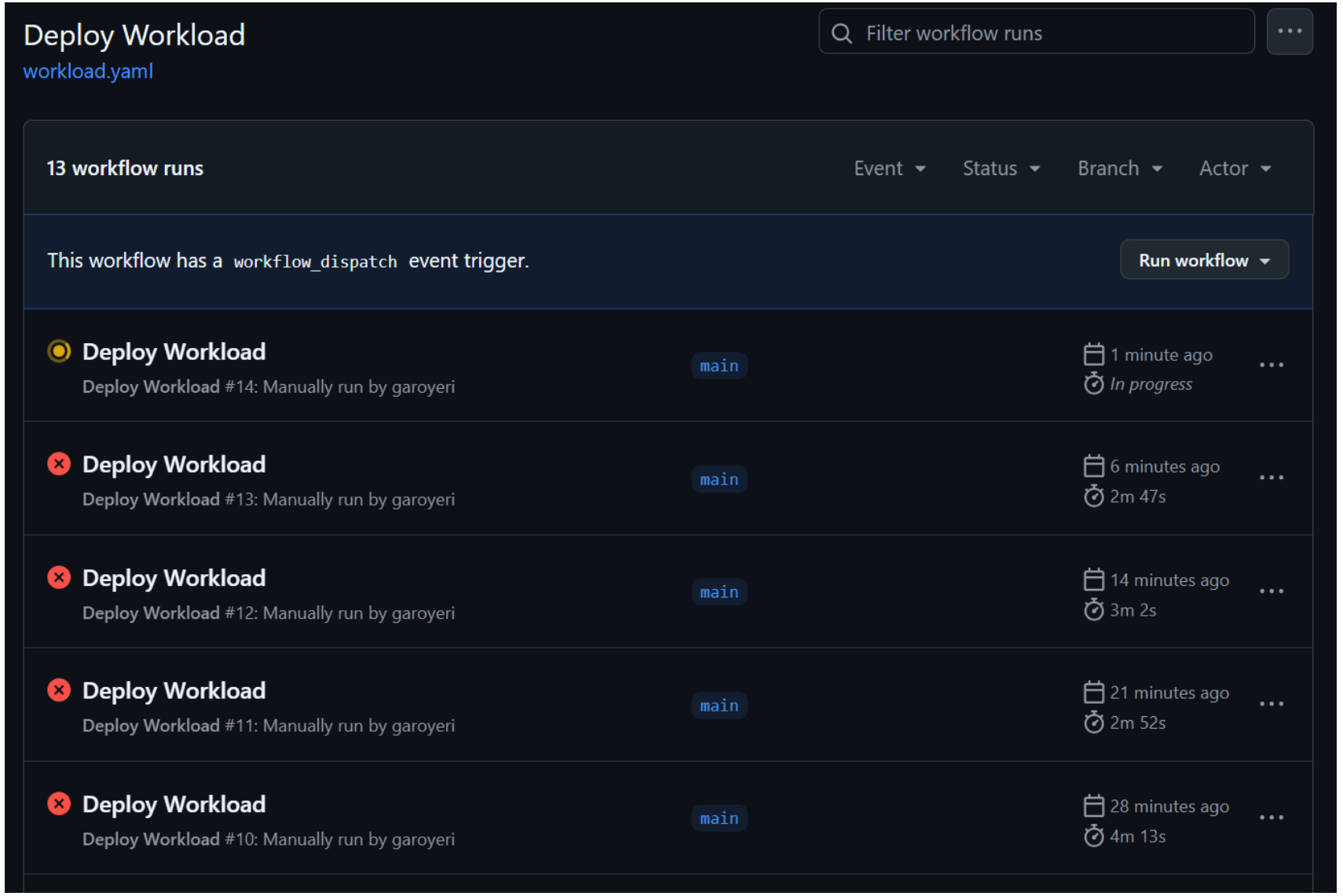
Development Team

- Use **declarative notation** for choosing infrastructure patterns (Bicep, Score, Terraform, whatever)
- **Talk** to platform team for requesting new infrastructure capabilities

Naming Things is Hard: You need a STRATEGY

- Example: Develop your naming and tagging strategy for Azure resources - Cloud Adoption Framework | Microsoft Learn
- Be picky on the name structures because once you start it's hard to fix (there's usually no way to rename things)
- If you can, enforce the naming strategy to avoid people making bad choices
- Be explicit about naming by using exported Bicep functions, types, and variables
- Make it easy to adopt the strategy and predict the names of things
- Follow the rules for naming: Naming rules and restrictions for Azure resources - Azure Resource Manager | Microsoft Learn
- Also: establish a clear tagging strategy and follow it!

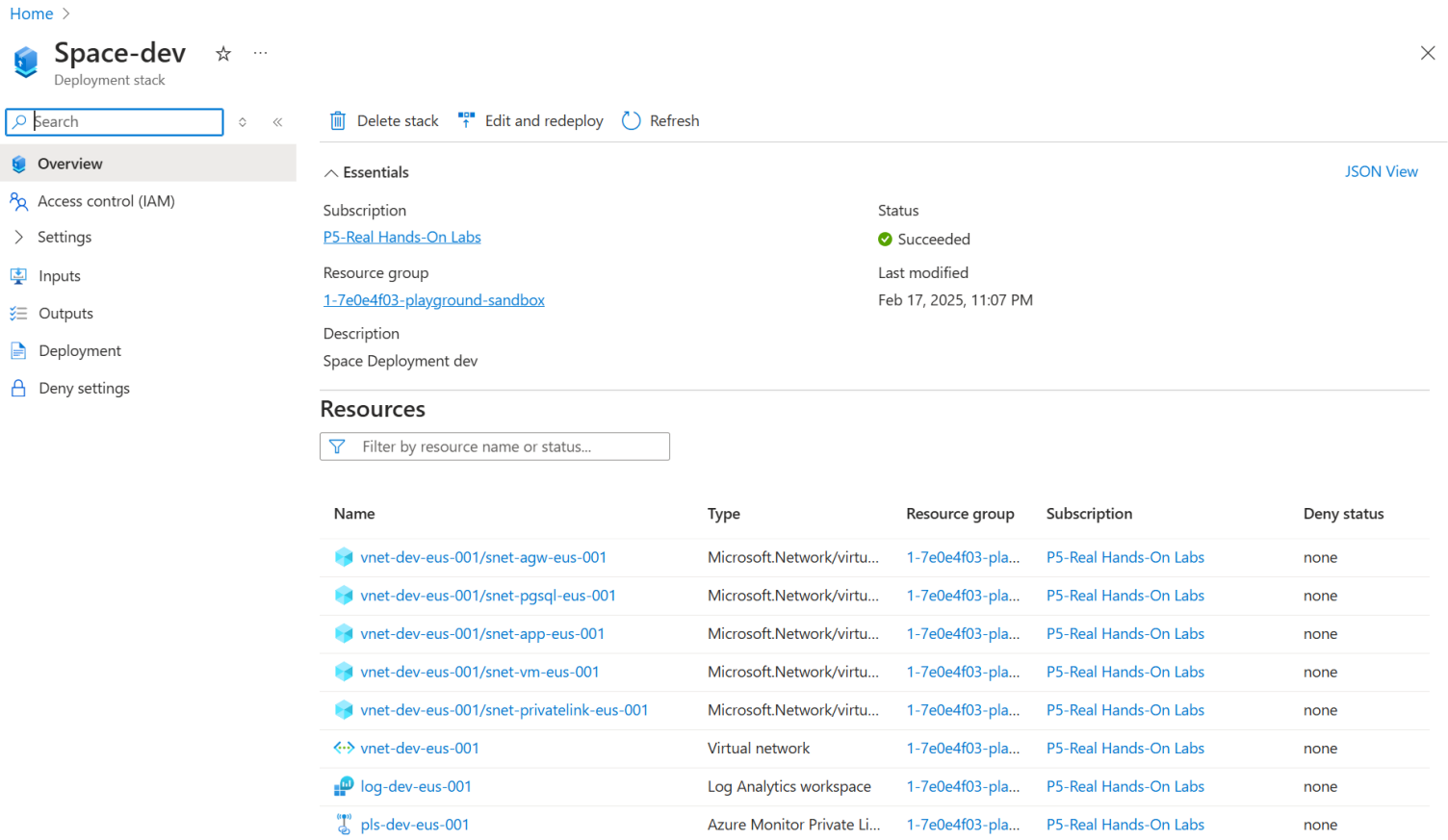
Figuring out Errors SUCKS.



Validating each change takes about 3-5 minutes and I get ONE error message each time.

😭

New Deployment Groups (Stacks?) Feature



- Does not support “whatif” but that’s probably ok (it wasn’t always that helpful)
- Allows for deletion of resources that don’t appear in Bicep scripts (or detach)
- Can add “deny” rules to block deletion of stacks until “deny” block is removed (doesn’t replace “locks” though).
- Great for developer iterations: bring up an environment, test it, tear it down without bothering others (deleting a stack deletes those resources).

Next Steps

- These examples are an OK starting point to get you familiar with how to use Bicep. Your actual Azure environment will be different (this demo has limitations due to the sandbox provided by ACloudGuru.
- Secrets management is a big concern: there are some automations you can do with Key Vault or consider a different 3rd party service with APIs and stuff.
  - You can roll your own secrets rotation with Key Vault and GitHub Actions, but it requires a lot of manual work
  - Hashicorp Vault and CyberArk Conjur provide a framework for building best practices around secrets rotation (but can be complicated to setup correctly)
- Establish a Platform Team who can focus on this stuff. It's HARD and COMPLICATED and FRUSTRATING.
  - Give them time and money
  - Consider 3rd party tools like Harness and Pulumi that give you more complete solutions over plain Bicep + ARM
  - Allow teams some bandwidth to fail and experiment to get to something that works for you
- Consider joining the Bicep Community Calls: https://aka.ms/armnews (they are super friendly and are showing new stuff each month)
  - Watch for AVM Community Calls: Community Calls | AVM

Thank You