😊 Azure Resource Manager

Azure Resource Manager (ARM) is the deployment and management service provided by Microsoft Azure for organizing and managing Azure resources. It enables you to provision, deploy, update, and manage resources in Azure in a consistent manner. With ARM, you can manage all the resources for your solution as a group, rather than individually.

Here are some key features and concepts related to Azure Resource Manager:

1. **Resource Groups:** ARM allows you to logically organize resources into groups called resource groups. Resource groups are containers that hold related resources for an Azure solution. They allow you to manage and apply access control, policies, and tags to a set of resources.
2. **Templates:** ARM templates are JSON files that define the resources to be deployed and their configuration. These templates allow you to describe your desired Azure infrastructure in a declarative way, making it easy to replicate and automate deployments.
3. **Deployment:** ARM enables you to deploy resources using various methods such as Azure Portal, Azure CLI, PowerShell, or through APIs. When you deploy a template, ARM handles the provisioning and configuration of the specified resources according to the template.
4. **Resource Providers:** Azure Resource Manager uses resource providers to manage resources. Each Azure service has its own resource provider responsible for managing the lifecycle of its resources.
5. **Role-Based Access Control (RBAC):** ARM integrates with Azure RBAC, allowing you to assign fine-grained permissions to users, groups, or applications for managing Azure resources.
6. **Tags:** Tags are key-value pairs that you can apply to resources to logically organize and categorize them. Tags can be used for cost tracking, resource management, and automation.

😄 Use Cases of Azure Resource Manager:

Azure Resource Manager (ARM) is a versatile tool with numerous use cases across various scenarios. Here are some common use cases:

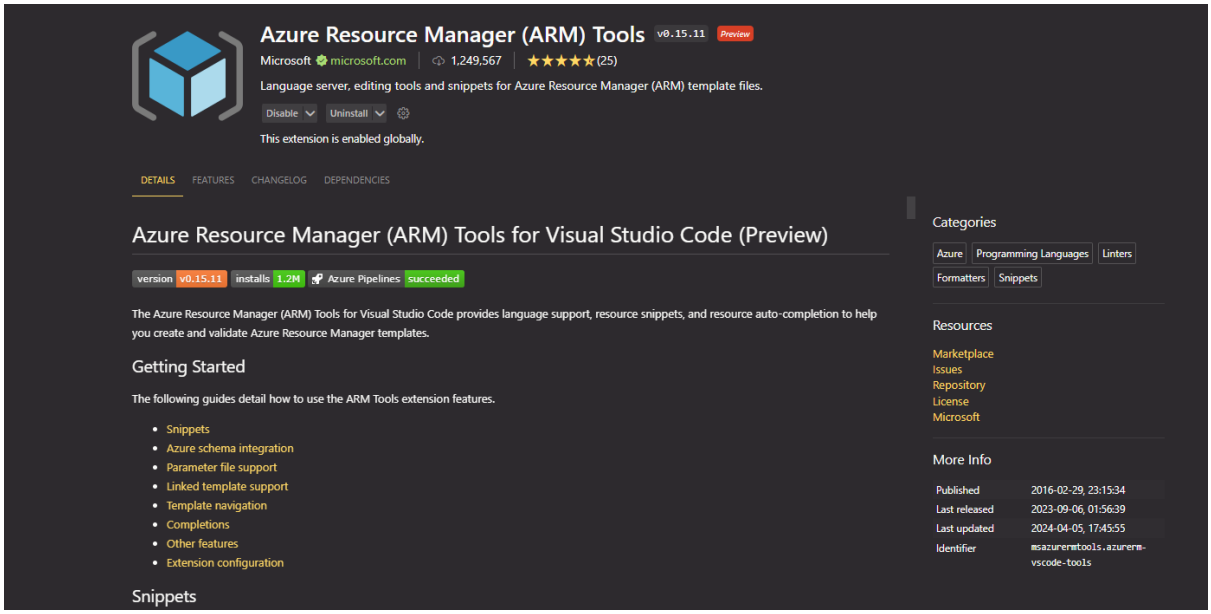
1. **Infrastructure as Code (IaC):** ARM templates enable Infrastructure as Code practices, allowing you to define your Azure infrastructure in a declarative manner. This facilitates automation, versioning, and consistency in deploying and managing infrastructure.
2. **Application Deployment:** ARM can be used to deploy and manage applications in Azure. You can define the required resources such as virtual machines, databases, storage accounts, and networking components in a template and deploy them as a single unit.

3. **DevOps and Continuous Integration/Continuous Deployment (CI/CD):** ARM integrates seamlessly with DevOps practices, enabling automated deployment pipelines. You can use ARM templates within CI/CD pipelines to provision and configure infrastructure as part of the deployment process.
4. **Resource Management and Governance:** ARM allows you to organize resources into resource groups and apply tags, policies, and RBAC permissions to manage and govern resources effectively. This helps in maintaining compliance, cost tracking, and enforcing organizational standards.
5. **Scalability and Elasticity:** ARM enables you to scale resources up or down based on demand by modifying the template parameters or using Azure Automation. This is particularly useful for handling variable workloads and optimizing resource usage.
6. **Disaster Recovery and High Availability:** ARM facilitates the setup of disaster recovery (DR) and high availability (HA) solutions by deploying resources across multiple Azure regions or availability zones. You can use templates to define failover configurations and automate the recovery process.
7. **Testing and Development Environments:** ARM can be used to provision and tear down testing and development environments quickly and consistently. Templates can define the entire environment, allowing teams to spin up isolated environments for testing new features or conducting experiments.
8. **Hybrid Cloud Scenarios:** ARM can manage resources not only in Azure but also in hybrid cloud environments. You can use Azure Arc to extend ARM capabilities to on-premises or multi-cloud environments, providing a unified management experience.

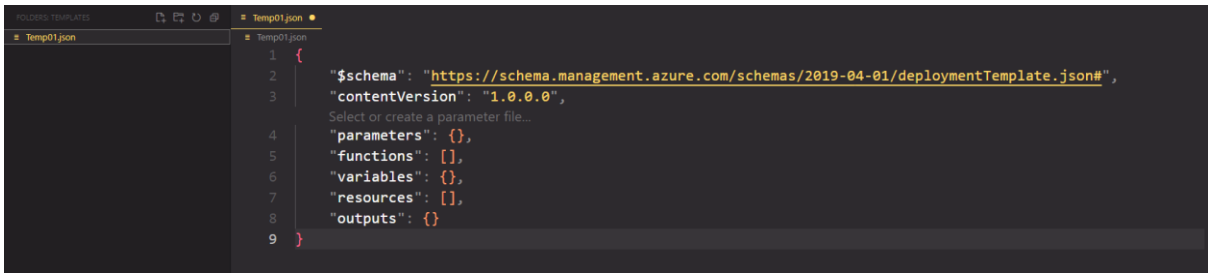
In this lab, we're learning how to create and deploy an Azure Resource Manager (ARM) template for provisioning a storage account in Microsoft Azure. The end goal is to understand how ARM templates facilitate Infrastructure as Code (IaC) practices, allowing us to define our Azure infrastructure in a declarative manner and automate the deployment process. By following the provided steps, we'll gain hands-on experience in using Visual Studio Code and PowerShell to create, modify, and deploy ARM templates, ultimately achieving a deeper understanding of Azure resource management and automation capabilities.

😄 To begin with the Lab:

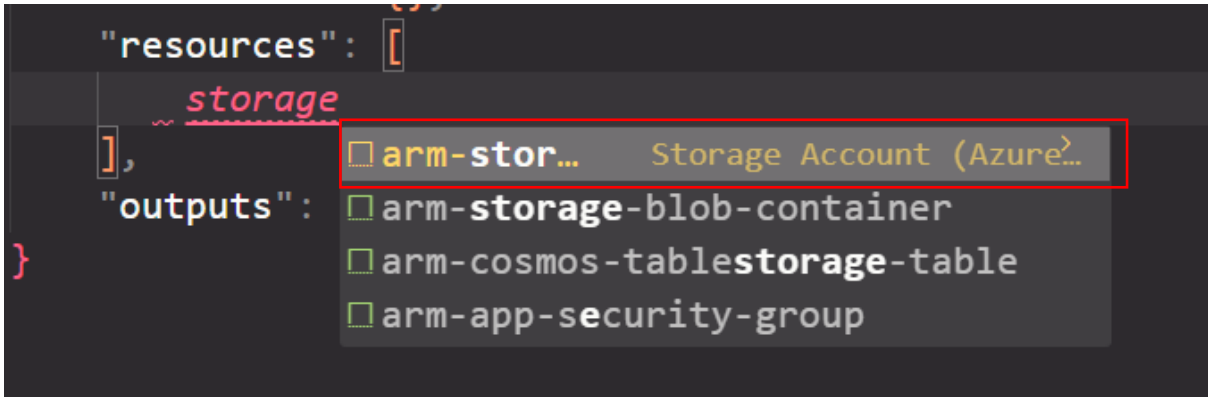
1. There are some pre-requisites for this lab: First, you should have Visual Studio Code installed in your laptop. Then you must install an extension for ARM in VS Code.



2. Now you have to create a new folder in any of your drive then open that empty folder in VS Code.
3. After that you are going to create a new file using VS Code.
4. And here simply if you will type **arm** and press enter then you will get the basic format of how you are going to write these templates.



5. Now if you expand resources and write storage in it then you will see that is suggesting you code. Now just click on it.



6. Then you will see that you got the code on how you will create your first storage account using ARM.
7. So, this is the default script which VS Code has given to us.
8. Now we are going to modify it.

```
"resources": [
    {
        "name": "storageaccount1",
        "type": "Microsoft.Storage/storageAccounts",
        "apiVersion": "2023-01-01",
        "tags": {
            "displayName": "storageaccount1"
        },
        "location": "[resourceGroup().location]",
        "kind": "StorageV2",
        "sku": {
            "name": "Premium_LRS",
            "tier": "Premium"
        }
    }
],
```

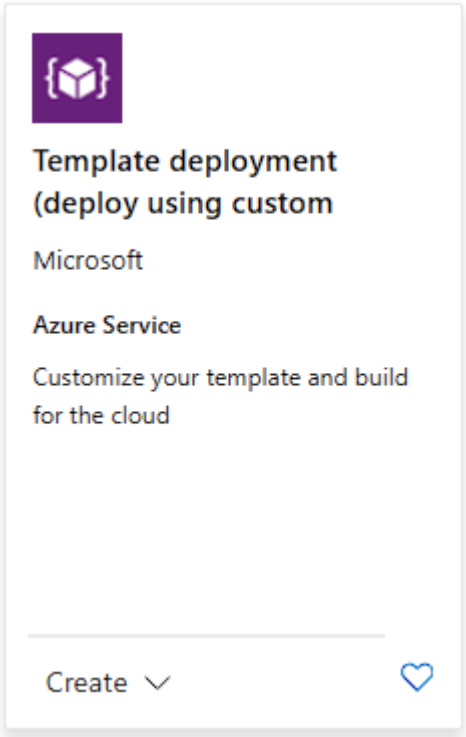
7. Now here you can see that in the resources part we have given our storage a unique name then the location where we want to create it.
8. After that we have changed the performance to standard and the redundancy to LRS (Locally Redundant Storage)
9. Now just save your template.

Temp01.json template01.json

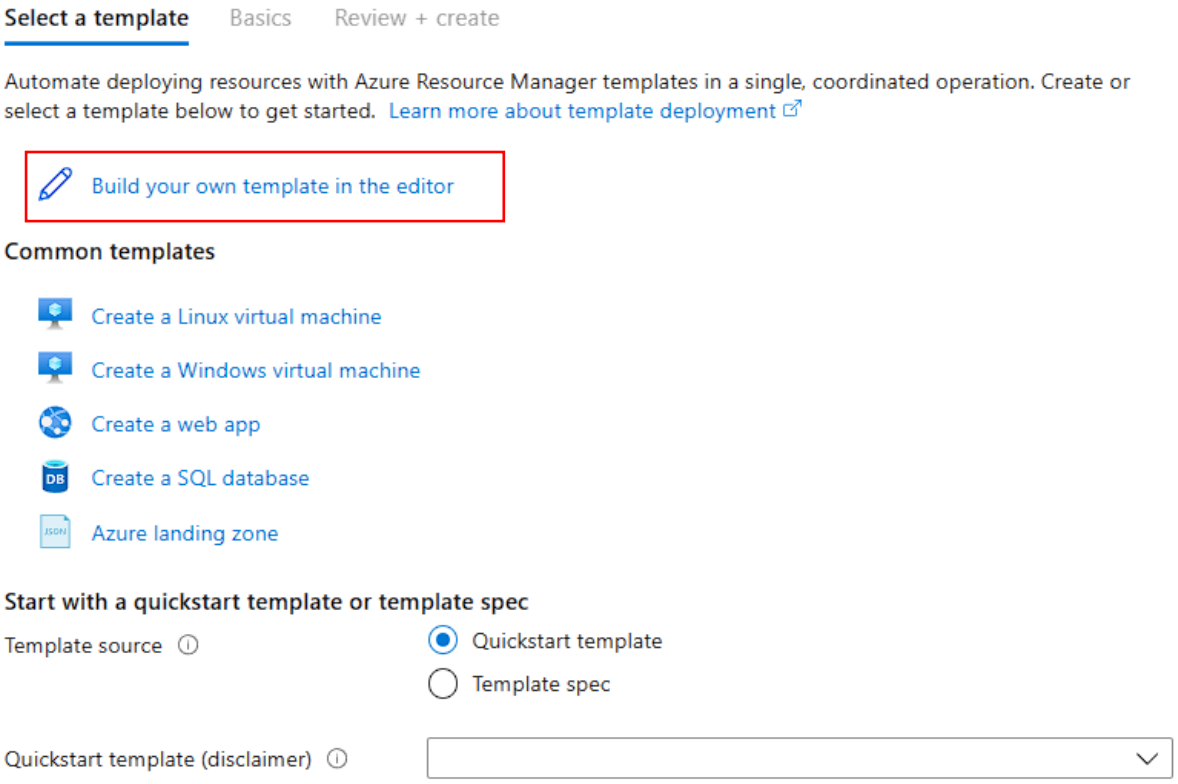
Temp01.json

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    Select or create a parameter file...
    "parameters": {},
    "functions": [],
    "variables": {},
    "resources": [
        {
            "name": "demotemp010101",
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "2023-01-01",
            "location": "Central India",
            "kind": "StorageV2",
            "sku": {
                "name": "Standard_LRS"
            }
        }
    ],
    "outputs": {}
}
```

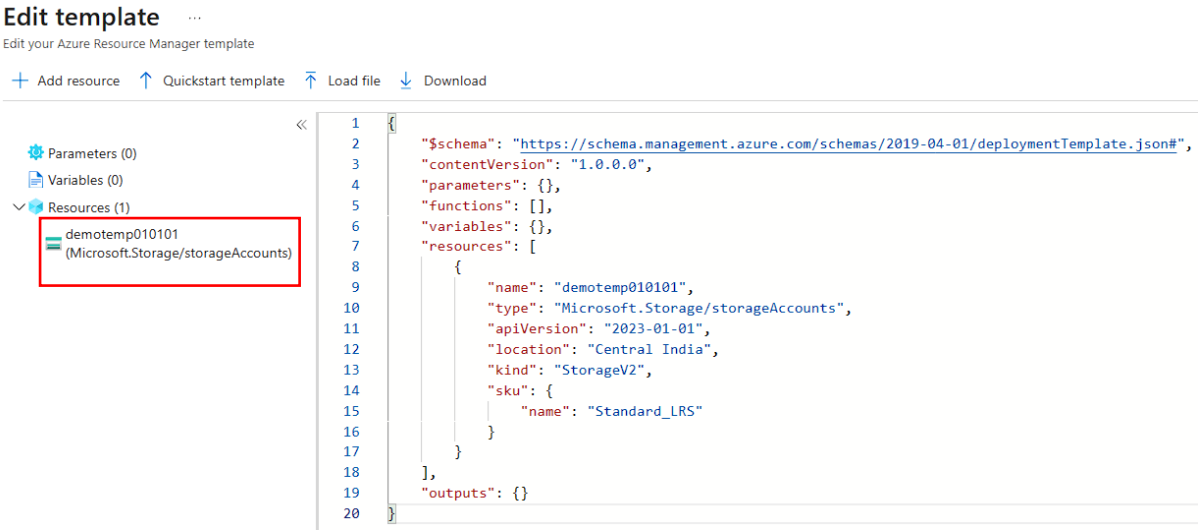
10. Now navigate to Azure Portal and from create resources search for template deployment and choose this service accordingly.



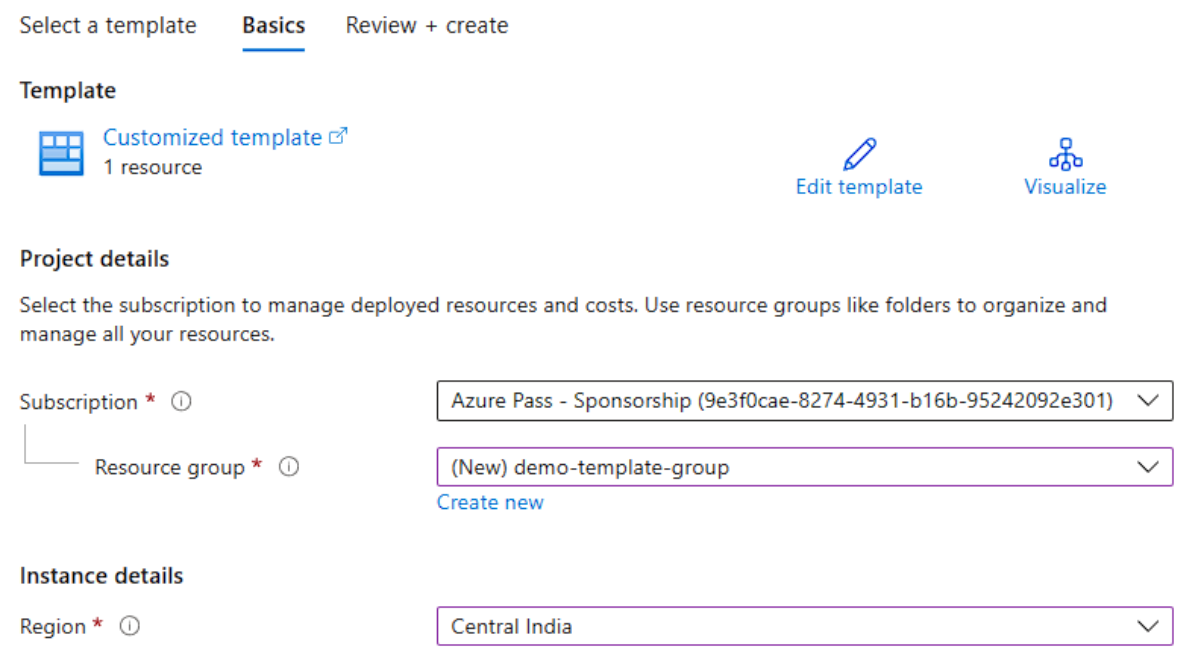
11. Now here you have to choose build your own template.



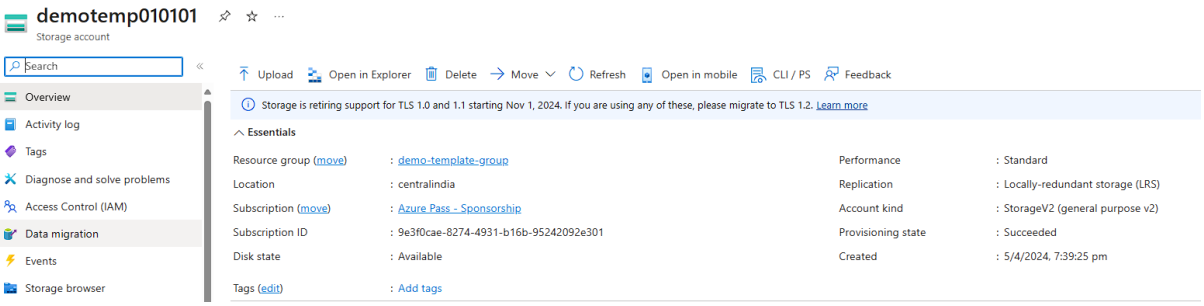
12. Then in the edit template section paste your JSON code here then on the left side you can see that your storage will be created.
13. Then just click on save.



14. After that choose your resource group or you can create a new one.
15. Then choose your region.



16. After that move to the review page and deploy your template.
17. Once your deployment is complete click on go to resources and it will take you directly to the storage account.
18. Below you can see your storage account in place.



19. Now you can also deploy your storage account using PowerShell.
20. In your VS Code open the terminal.
21. Now you can make one change which is to change the name of your storage account because it should be unique.
22. Then you are going to install Azure CLI if not yet installed. Search for Download Azure CLI on Google and then download it from the official Microsoft page.
23. Now you are going to connect your account using PowerShell.

    Connect-AzAccount

```
PS C:\Windows\system32> Connect-AzAccount
WARNING: Both Az and AzureRM modules were detected on this machine. Az and AzureRM modules cannot be imported in the
same session or used in the same script or runbook. If you are running PowerShell in an environment you control you c
an
 use the 'Uninstall-AzureRm' cmdlet to remove all AzureRm modules from your machine. If you are running in Azure Auto
mation, take care that none of your runbooks import both Az and AzureRM modules. More information can be found here:
https://aka.ms/azps-migration-guide

Account                SubscriptionName         TenantId                             Environment
-------                ----------------         --------                             -----------
behal.ritesh@gmail.com Azure Pass - Sponsorship 3d18f3ae-8875-4771-aaa9-a9afcd43d751 AzureCloud


PS C:\Windows\system32>
```

24. Once your account is connected. Then you are going to issue this command.
25. In this command we are defining that it is a new deployment, and our resource group name is this and our template file name is this.
26. After that just run the command.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  CODE REFERENCE LOG

PS D:\Azure Resource Manager\Templates> New-AzResourceGroupDeployment -ResourceGroupName demo-template-group -TemplateFile Temp01.json
```

27. Below you can see that our command was successfully executed.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  CODE REFERENCE LOG                    powershell

PS D:\Azure Resource Manager\Templates> New-AzResourceGroupDeployment -ResourceGroupName demo-template-group -TemplateFile Temp01.json
WARNING: Both Az and AzureRM modules were detected on this machine. Az and AzureRM modules cannot be imported in the same session or used in the same script or runbook. If you are running PowerShell in an
environment you control you can use the 'Uninstall-AzureRm' cmdlet to remove all AzureRm modules from your machine. If you are running in Azure Automation, take care that none of your runbooks import both Az and
AzureRM modules. More information can be found here: https://aka.ms/azps-migration-guide


DeploymentName          : Temp01
ResourceGroupName       : demo-template-group
ProvisioningState       : Succeeded
Timestamp               : 05-04-2024 14:41:23
Mode                    : Incremental
TemplateLink            :
Parameters              :
Outputs                 :
DeploymentDebugLogLevel :


PS D:\Azure Resource Manager\Templates>
```

28. Now you have to navigate to Azure Portal then towards storage account.
29. And there you will see your storage account.