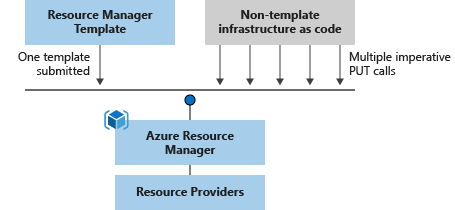
**What are ARM templates?**

ARM Templates are a way to declare the objects you want, the types, names and properties in a JSON file which can be checked into source control and managed like any other code file. ARM Templates are what really gives us the ability to roll out Azure “Infrastructure as code”.

**Why we choose ARM Templates?**

* **Declarative syntax**: ARM templates allow you to create and deploy an entire Azure infrastructure declaratively. For example, you can deploy not only virtual machines, but also the network infrastructure, storage systems and any other resources you may need.
* **Repeatable results**: Repeatedly deploy your infrastructure throughout the development lifecycle and have confidence your resources are deployed in a consistent manner. Templates are idempotent, which means you can deploy the same template many times and get the same resource types in the same state
* **Orchestration**: You don't have to worry about the complexities of ordering operations. Resource Manager orchestrates the deployment of interdependent resources so they're created in the correct order. When possible, Resource Manager deploys resources in parallel so your deployments finish faster than serial deployments.



* **CI/CD integration**: You can integrate templates into your continuous integration and continuous deployment (CI/CD) tools, which can automate your release pipelines for fast and reliable application and infrastructure updates. By using Azure DevOps and Resource Manager template task, you can use Azure Pipelines to continuously build and deploy ARM template projects

**Template file**

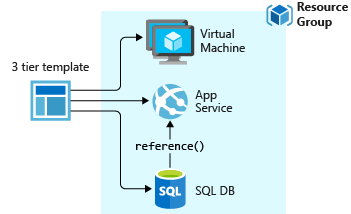
Within your template, you can write template expressions that extend the capabilities of JSON. These expressions make use of the functions provided by Resource Manager.

The template has the following sections:

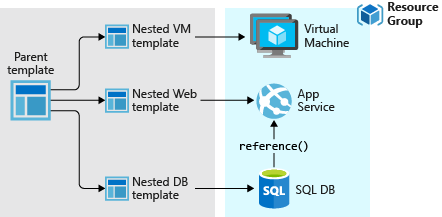
* Parameters- Provide values during deployment that allow the same template to be used with different environments.
* Variables- Define values that are reused in your templates. They can be constructed from parameter values.
* User-Defined functions- Create customized functions that simplify your template.
* Resources- Specify the resources to deploy.
* Outputs- Return values from the deployed resources.

**Template Design:**

For example, you can deploy your three tier application through a single template to a single resource group.



But, you don't have to define your entire infrastructure in a single template. Often, it makes sense to divide your deployment requirements into a set of targeted, purpose-specific templates. You can easily reuse these templates for different solutions.



**Creation of Template:**

**Deployment of Template:**