

Expected Learning Outcomes

Azure Masterclass: Understanding the ARM Template Components



By the end of this section, you should be able to:

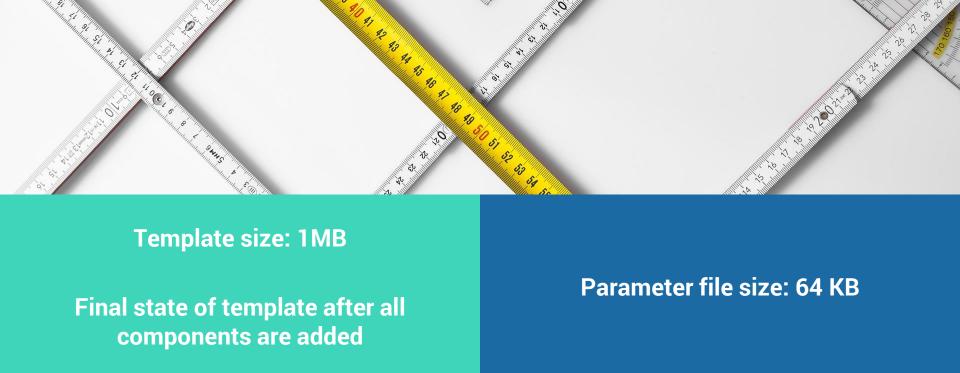
- **→** Define template limits for the ARM components
- Describe resource name limitations
- **Explain best practices for using ARM components**
- Define nested templates and how they are used

Recap: Structure of an ARM Template

Azure Masterclass: Understanding the ARM Template Components

Here is the basic structure of an ARM template:

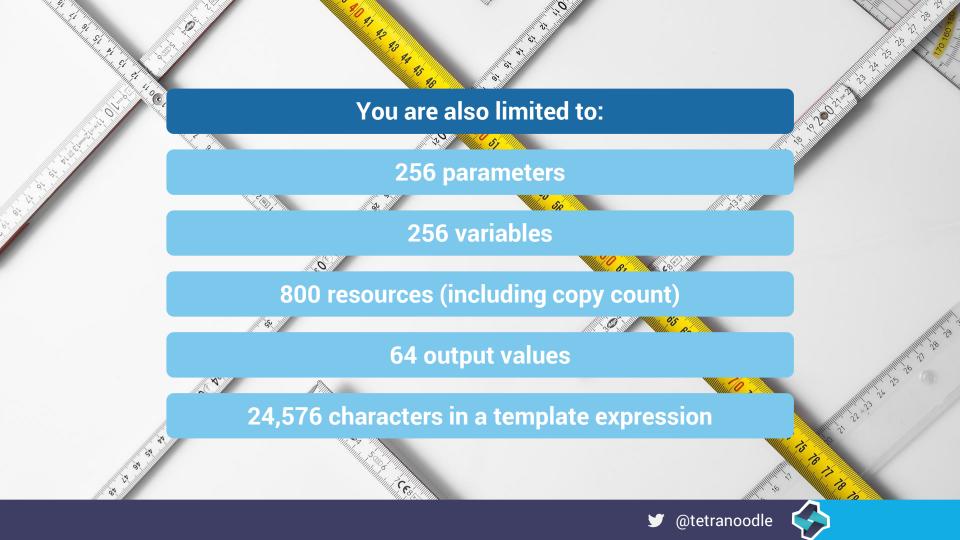
```
DeploymentTemplate.json*
http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#
      ∃{
          "$schema": "http://schema.management.azure.com/schemas/2014-04-01-preview/deploymentTemplate.json#",
          "contentVersion": "<version-number-of-template>",
          "parameters": { <parameter-definitions-of-template> },
          "variables": { <variable-definitions-of-template> },
          "resources": [ { <definition-of-resource-to-deploy> } ],
          "outputs": { <output-of-template> }
```











Resource Name Limitations

Azure Masterclass: Understanding the ARM Template Components

→ There are three types of resource names in Resource Manager:











Parameters: Best Practices

Azure Masterclass: Understanding the ARM Template Components

There are two strategies to having parameters in ARM templates:



User is technically equip



Would you use parameters differently for these two users?

Scenario 2:

is not technically equipped



Should have many parameters
Allows user to customize template

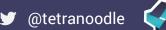
Use minimum parameters
Develop a marketplace offering



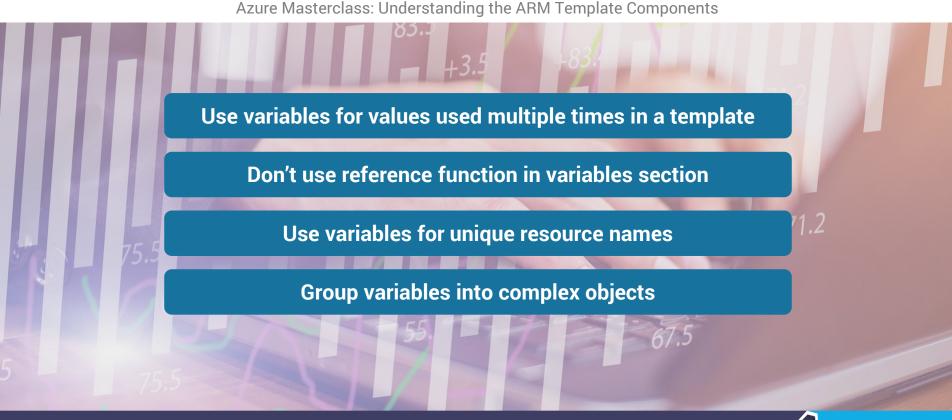


- Minimize use of parameters
- Use camel case for parameter names
- → Add description in metadata
- Define default values
- Use SecureString
- Don't use parameters to specify location
- Avoid parameter for API version



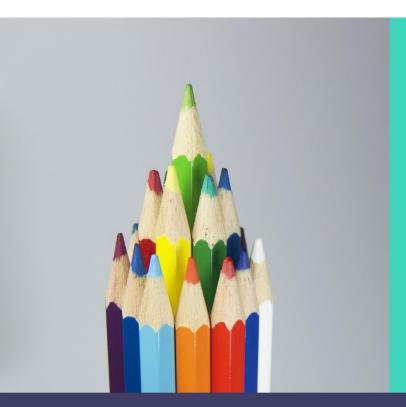


Variables: Best Practices



Resources: Best Practices

Azure Masterclass: Understanding the ARM Template Components



- → Add resource metadata using tags
- → Do not hard-code namespace
- Comments for each resource
- → No namespace if storage account in same template
- Public namespace value reflects same function
- Public IP addresses only when required
- Unique DominNameLabel property
 - ^[a-z][a-z0-9-]{1,61}[a-z0-9]\$



Outputs: Best Practices

Azure Masterclass: Understanding the ARM Template Components

- **→** Templates for public IP addresses should include an outputs section that returns IP address details
- Retrieve details about public IP addresses and FQDNs after deployment
- Use the API version used to create the resource when referencing it





Break down a solution into targeted components

Can have different main templates

Use parameters to conditionally link to nested templates



Useful Tips

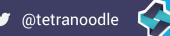
Azure Masterclass: Understanding the ARM Template Components

Pass templates through JSON validator

Format JSON for better readability











Quick Start Templates

















