



ARM Templates: Structure, Scopes and Modes

Microsoft Learn: ARM template reference, template functions, deployment modes, template specs, deployment scripts, as of 2026 | The JSON engine under Bicep: sections, expressions, four scopes, Incremental vs Complete, linked templates, template specs

Azure ARM JSON Template specs AZ-104 AZ-204

Anatomy of a template

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "prefix": {
      "type": "string",
      "maxLength": 11
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "variables": {
    "name": "[format('{0}{1}', parameters('prefix'), uniqueString(resourceGroup().id))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2023-05-01",
      "name": "[variables('name')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_ZRS"
      },
      "kind": "StorageV2",
      "properties": {
        "minimumTlsVersion": "TLS1_2",
        "allowBlobPublicAccess": false
      }
    }
  ],
  "outputs": {
    "blob": {
      "type": "string",
      "value": "[reference(variables('name')).primaryEndpoints.blob]"
    }
  }
}
```

your version; the \$schema above sets the scope (RG here)

prefix + 13-char hash

pins the property schema

reference() reads runtime state

Section	Required	What goes in	Limit
\$schema	yes	schema URL; sets the scope and the editor's validation	-
contentVersion	yes	your own version string, e.g. 1.0.0.0	-
languageVersion	no	2.0 enables symbolic resource names and user-defined types	-

parameters	no	inputs: type, defaultValue, allowedValues, min/max, meta-data	256
variables	no	computed values reused in the template	256
functions	no	user-defined functions in a namespace	-
resources	yes	type, apiVersion, name, location, properties, dependsOn, copy, condition	800 (copies count)
outputs	no	values returned to the caller or pipeline	64

INFO

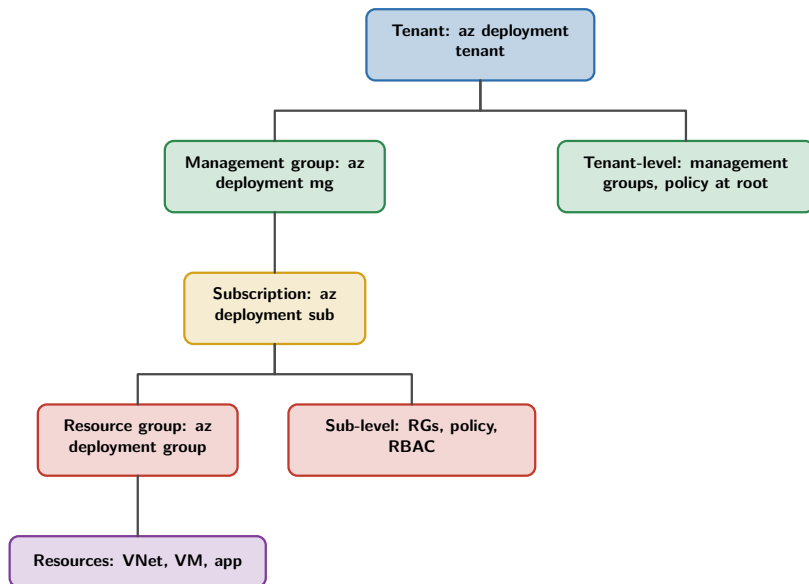
Other limits that bite: template size 4 MB after expansion, parameter values file 4 MB, expression up to 24,576 characters, 800 deployments in a resource group's history (ARM deletes the oldest automatically when close to the limit). Parameter types: string, securestring, int, bool, object, secureObject, array.

Functions you use every day

Function	Returns / use
resourceGroup().location, subscription().subscriptionId	deployment context; tenant(), managementGroup(), deployment().name, environment().suffixes.storage
resourceId('Microsoft.Network/virtualNetworks/subnets', vnet, 'app')	full ID of a resource (child types take one name per level); subscriptionResourceId, tenantResourceId, extensionResourceId for other scopes
reference(name, apiVersion, 'Full')	runtime properties; 'Full' adds identity, sku, tags; implicit dependency when the resource is in the same template
listKeys(resourceId(...), '2023-05-01').keys[0].value	list* actions (listKeys, listSecrets, listConnectionStrings); avoid in outputs, they leak secrets
uniqueString(resourceGroup().id), guid(a, b)	13-char hash, deterministic per input; guid() for role-assignment names
copyIndex(), length(), range(), union(), contains()	loops and collections; if(cond, a, b), coalesce(), empty()
utcNow(), newGuid()	only allowed in a parameter defaultValue (non-deterministic)
format('{0}-{1}', a, b), concat(), toLower(), take()	string building; take(name, 24) respects storage name length

reference() runtime state of a deployed resource	vs	resourceId() string ID built from names
Tell: Do you need a value that only exists after creation (endpoint, principalId)?		
Incremental mode adds and updates, never deletes	vs	Complete mode deletes what the template omits
Tell: Will resources missing from the file survive?		
Nested template inline JSON in the parent	vs	Linked template separate file by URI or relativePath
Tell: Is the child template text inside the parent file?		
Template spec versioned Azure resource, RBAC	vs	Linked template in a storage account URL plus SAS token
Tell: Is access granted by role or by token?		
securestring parameter value never logged in deployment history	vs	Key Vault reference parameter file points at a secret
Tell: Does the secret ever leave Key Vault in clear text?		

Scopes and how to deploy



Each scope has its own schema and command; a template can target a child scope with nested deployments

az

deployment

group

create

-g

<RG>

-f

<FILE>

[-p PARAMS...]

[-n NAME]

[--mode Incremental|Complete]

[-c]

[--what-if]

command = type exactly

literal = type exactly

<PLACEHOLDER> = you supply a value

[optional] = may be omitted

RG --- target resource group (exists already)

FILE --- azuredeploy.json or main.bicep (compiled on the fly)

PARAMS --- @params.json, main.bicepparam or key=value

NAME --- deployment name; default = file name, re-using it overwrites history

Scope	Command	Typical resources	Schema file
Resource group	az deployment group create -g rg	VNets, VMs, apps, databases	deploymentTemplate.json
Subscription	az deployment sub create -l westeurope	resource groups, policy and role assignments, budgets	subscriptionDeploymentTemplate.json
Management group	az deployment mg create -m mg-id -l westeurope	policy definitions and assignments, role definitions	managementGroupDeploymentTemplate.json
Tenant	az deployment tenant create -l westeurope	management groups, subscriptions (aliases)	tenantDeploymentTemplate.json

EXAM TIP

Scopes above the resource group need --location (-l): ARM stores the deployment meta-data there. Complete mode exists only at resource-group scope. PowerShell equivalents: New-AzResourceGroupDeployment, New-AzSubscriptionDeployment (alias New-AzDeployment), New-AzManagementGroupDeployment, New-AzTenantDeployment.

Modes, loops, conditions, dependencies

copy loop, condition and explicit dependency

```

{
  "type": "Microsoft.Network/publicIPAddresses",
  "apiVersion": "2024-05-01",
  "name": "[format('pip-web-{0}', copyIndex(1))]",
  "condition": "[parameters('public')]",
  "copy": { "name": "pips", "count": "[parameters('n')]",
            "mode": "Serial", "batchSize": 2 },
  "location": "[parameters('location')]",
  "sku": { "name": "Standard" },
  "properties": { "publicIPAllocationMethod": "Static" },
  "dependsOn": [
    "[resourceId('Microsoft.Network/virtualNetworks', 'vnet')]"
  ]
}

```

- **copy** count 0 to 800 instances; mode: Serial + batchSize deploys in waves; also on properties (copy inside properties), variables and outputs
- **condition** false skips the resource; with Complete mode and current API versions, a skipped resource that exists is deleted

- **dependsOn** only inside the same template; ARM parallelises the rest, so missing dependencies cause random ResourceNotFound
- **Complete mode** deletes unlisted resources in the RG (respecting locks); nested templates always run Incremental. Microsoft now recommends deployment stacks instead
- **Linked templates** templateLink.uri must be reachable by ARM (public URL or SAS); relativePath resolves from the parent's URI or template spec; expression EvaluationOptions.scope: inner isolates nested parameters

PROBLEM One template must create 3 environments × 4 storage accounts named from a prefix. Check the 800-resource limit and the 24-character storage-name limit.

GIVEN

- prefix stdata (6 chars)
- uniqueString() = 13 chars
- env suffix t/s/p + index 1-4 = 2 chars
- storage name: 3-24 lowercase letters and digits

1 **Count resources after expansion**

3 x 4 = 12 copies (copy count counts)

Why: the 800 limit applies to expanded instances

2 **Compute the name length**

6 + 13 + 2 = 21 characters

Why: 21 <= 24, no take() needed

3 **Make names deterministic**

uniqueString(resourceGroup().id) per RG

Why: same RG -> same name on every rerun (idempotent)

4 **Lowercase**

toLower(format('{0}{1}{2}{3}', ...))

Why: uniqueString is lowercase but prefixes may not be

ANSWER 12 resources, names of 21 characters: within both limits

CHECK ✓ A 12-character prefix would give 27 characters: wrap in take(..., 24).

🔒 Template specs, scripts, secrets

Command	Effect
az ts create -n vnet -g rg-specs -v 1.2 -l westeurope -f vnet.bicep	publish version 1.2 (Bicep compiled to JSON on publish)
az deployment group create -g rg-app --template-spec <ts-version-id>	deploy from the spec; caller needs Reader on the spec, rights on the target
az ts list -g rg-specs -o table / az ts show -n vnet -g rg-specs -v 1.2	inventory and inspect versions
az group export -n rg-app > exported.json	export current resources as a template (starting point, needs clean-up)
az bicep decompile -f azuredeploy.json	convert ARM JSON to Bicep (best effort, review warnings)

```
Key Vault reference in a parameters file
"adminPassword": {
  "reference": {
    "keyVault": { "id": "/subscriptions/<sub>/resourceGroups/rg-sec/providers/Microsoft.KeyVault/vaults/kv-prod" },
    "secretName": "vmAdminPassword"
  }
}
```

💡 **TIP**

The vault needs enabledForTemplateDeployment: true, and the deployer needs Microsoft.KeyVault/vaults/deploy/action (in Owner and Contributor, or a custom role). The value never appears in deployment history because the parameter is securestring.

📜 **Deployment script**

Microsoft.Resources/deploymentScripts (kind AzureCLI or AzurePowerShell) runs your script in Azure Container Instances with a storage account, under a user-assigned managed identity; retentionInterval (max 26 hours) and cleanupPreference control clean-up; outputs go to \$AZ_SCRIPTS_OUTPUT_PATH. Use it for gaps such as uploading a file or calling a data-plane API.

🚧 Pitfalls and recap

YOU MAY HEAR

"Removing a resource from the template deletes it"

BUSTED

✓ **ACTUALLY**

Not in Incremental mode (the default)

Only Complete mode or a deployment stack with delete actions removes it.

YOU MAY HEAR

"Incremental mode only adds the properties I list"

BUSTED

✓ **ACTUALLY**

It replaces the resource's properties with the template's

Unlisted properties may reset to defaults (classic trap: subnets dropped when a VNet is redeployed without them).

YOU MAY HEAR

"dependsOn across two templates works"

BUSTED

✓ **ACTUALLY**

dependsOn only sees resources of the same template

Chain the linked deployment resources instead, or pass outputs.

Tick a box only when you could explain the point to someone else

- | | |
|---|---|
| ☐ Schema decides the scope
four schemas, four az deployment sub-commands | ☐ Pin apiVersion
properties change between versions |
| ☐ uniqueString for names
deterministic, 13 chars | ☐ Incremental by default
Complete deletes; prefer stacks |
| ☐ Template specs over SAS URLs
versioned, RBAC-shared | ☐ Secrets via securestring + Key Vault
never in outputs |

Q1 Which two top-level elements are mandatory, besides resources?

A `$schema` and `contentVersion`.

Q2 Max resources per template, and does a copy loop count each instance?

A 800, and yes, every copied instance counts.

Q3 What does `uniqueString(resourceGroup().id)` return?

A A deterministic 13-character lowercase hash: same RG, same value on every deployment.

Q4 At which scopes is Complete mode allowed?

A Resource group only.

Q5 Which extra flag do subscription, management-group and tenant deployments need?

A `--location`, where the deployment metadata is stored.

Q6 Two functions only allowed in a parameter `defaultValue`?

A `utcNow()` and `newGuid()`.

Q7 How do you share a template with other teams without SAS URLs?

A A template spec (`az ts create`), shared by giving Reader on it.

Q8 Which setting must a Key Vault have before a parameter file can reference its secrets?

A `enabledForTemplateDeployment: true`.

Q9 What runs a CLI script during an ARM deployment?

A A `Microsoft.Resources/deploymentScripts` resource (ACI container with a managed identity).