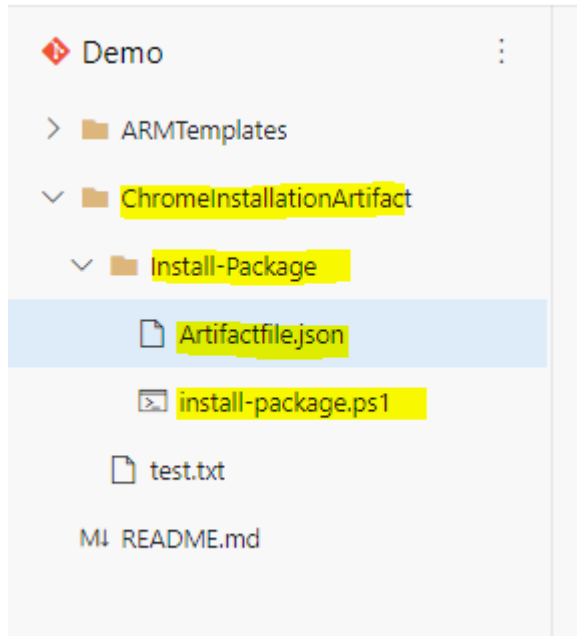


How to add an custom artifact for Dev Test Lab and deploy it with ARM template from pipeline

Step 1: Create an custom Artifact

- While creating an custom artifact create a folder with any name.
- Put the .ps1 script and an file "Artifactfile.json" as below. ([reference link](#))



- Configure Artifactfile.json to run the .ps1 file.
Sample Artifactfile.json will look like:

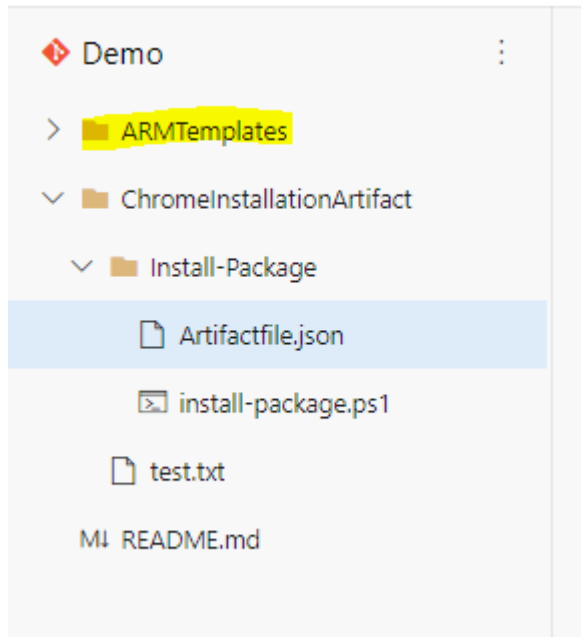
```
{
  "$schema": "https://raw.githubusercontent.com/Azure/azure-devtestlab/master/schemas/2016-11-28/dtlArtifacts.json",
  "title": "<<Expected title>>",
  "publisher": "Saumya Mishra",
  "description": "Description",
  "tags": [
    "Exchange"
  ],
  "targetOsType": "Windows",
  "parameters": {
    "pm1": {
      "type": "string",
      "displayName": "",
      "description": ""
    }
  },
}
```

```

"pm2": {
  "type": "string",
  "displayName": "",
  "description": ""
},
"pm3": {
  "type": "string",
  "displayName": "",
  "description": ""
},
"pm4": {
  "type": "securestring",
  "displayName": "PAT",
  "description": ""
}
},
"runCommand": {
  "commandToExecute": "[concat('powershell.exe -ExecutionPolicy bypass \"%& ./install-package.ps1', ' -pm1 ', parameters(pm1), ' -pm2 ', parameters(pm2), ' -pm3 ', parameters(pm3), ' -PAT ', parameters(pm4))]"
}
}

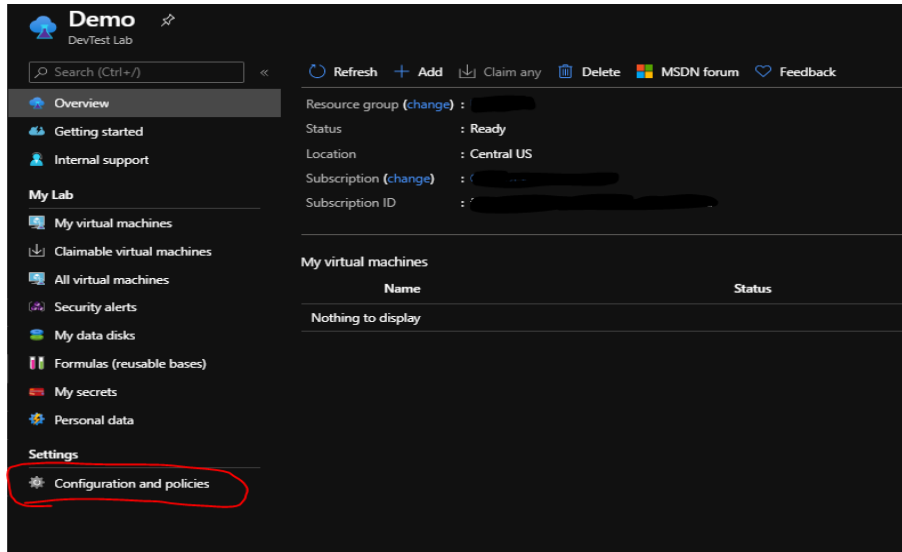
```

- Create an ARM template for which we are going to access the artifact.

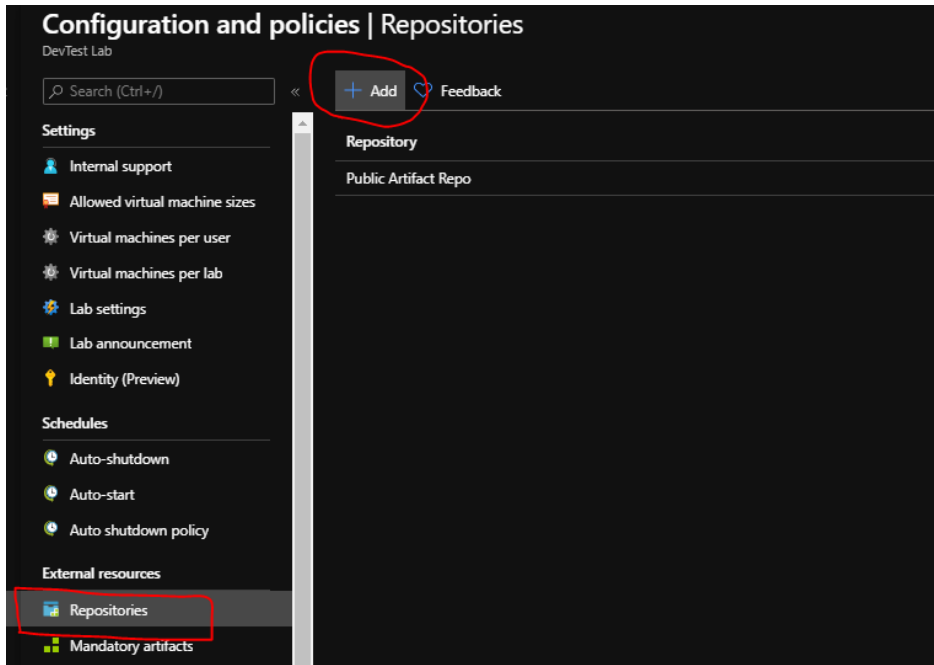


Step 2: Add the Artifact to DevTestLab Repository

- 1) Create a new DevTest Lab
- 2) Go to configuration and policies:



- 3) Add a repository.



4) Fill the form to create an artifact

The screenshot shows the 'Repository' configuration page in Azure DevOps. The breadcrumb navigation at the top is 'Home > Resource groups > [Resource Group] > Demo > Configuration and policies | Repositories >'. The form has the following fields and annotations:

- Name ***: A text box containing 'Demo'. A red line points from the text 'Name of the repo' to this field.
- Git clone URI ***: A text box containing 'https://dev.azure.com/SaumyaMishraProjects/_git/Demo'. A red line points from the text 'link to devops repo' to this field.
- Branch**: A text box containing 'master'. A red line points from the text 'Branch Name' to this field.
- Personal access token ***: A masked text box with dots. A red line points from the text 'PAT token' to this field.
- Folder paths**: A section header with the instruction 'At least one folder path is required below'.
 - Artifact folder path**: A text box containing '/ChromeInstallationArtifact'. A red line points from the text 'Artifact Folder' to this field.
 - Azure Resource Manager template folder path**: A text box containing '/ARMTemplates'. A red line points from the text 'ARM Template folder name' to this field.

Step 3 : Get the name (NOT display name) of the artifact.

- Goto [link](#) and run the command as below, the output of the API will give you the name of the Artifact.

The screenshot shows a terminal window with the following content:

```
List artifact sources in a given lab.  
  
HTTP  
GET https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DevTestLa
```

At the top right of the terminal, there is a 'Copy' button and a green button labeled 'Try It' which is circled in red.

- Login with your account and provide the details as below and you should be able to see the name of the artifact

REST API Try It

Try the REST API with the inputs below.

Sign out

Request URL

GET

https://management.azure.com/subscriptions/[REDACTED]/resourceGroups/[REDACTED]/pro

Parameters

labName*

[REDACTED]

resourceGroupName*

[REDACTED]

subscriptionId*

[REDACTED]

api-version*

2018-09-15

name

value



Sample output:

Body

JSON

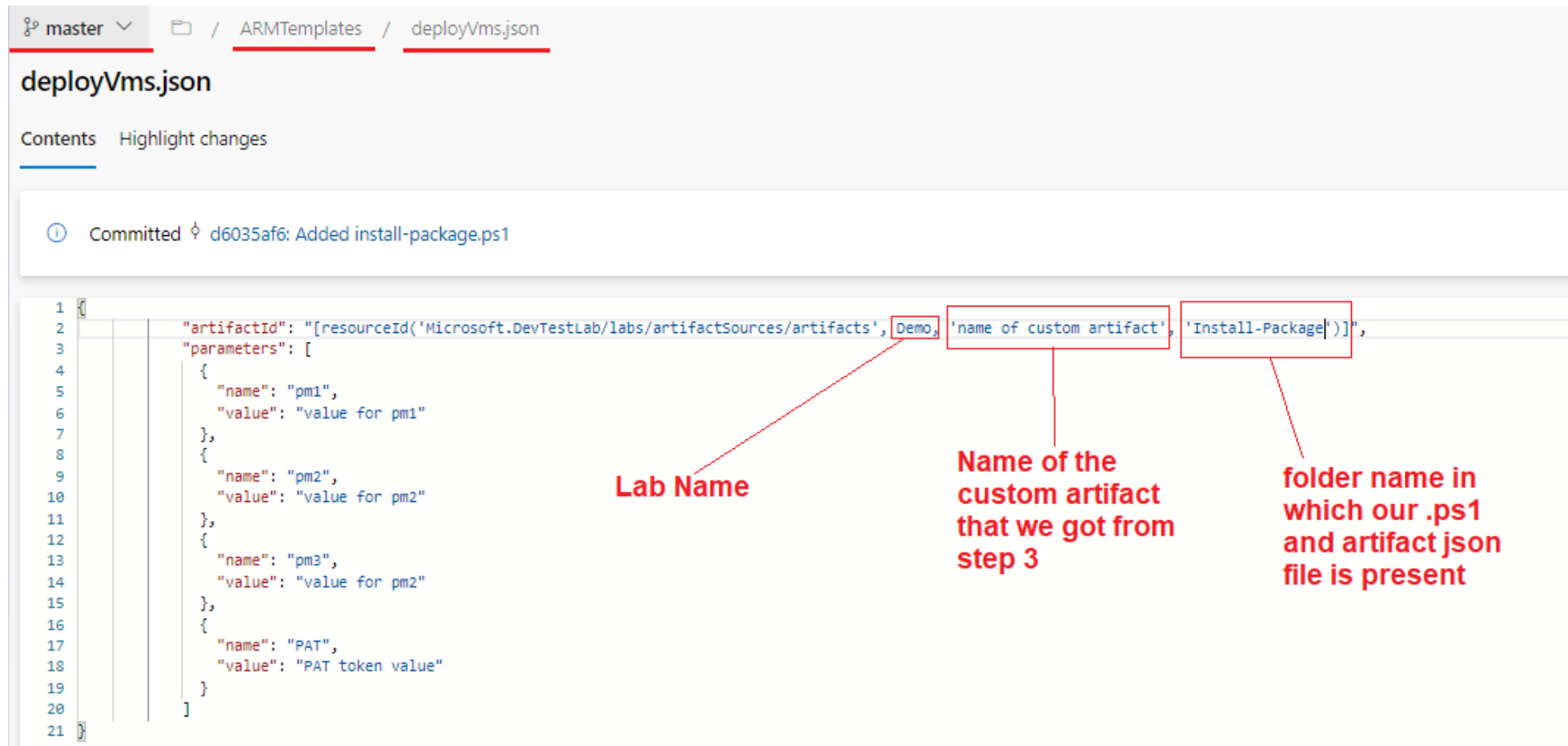
Copy

```
{
  "value": [
    {
      "properties": {
        "displayName": "[REDACTED]",
        "uri": "[REDACTED]",
        "sourceType": "VsoGit",
        "folderPath": "[REDACTED]",
        "armTemplateFolderPath": "[REDACTED]",
        "branchRef": "[REDACTED]",
        "status": "Enabled",
        "createdDate": "[REDACTED]",
        "provisioningState": "Succeeded",
        "uniqueIdentifier": "[REDACTED]"
      },
      "id": "[REDACTED]",
      "name": "[REDACTED]",
      "type": "[REDACTED]"
    }
  ]
}
```

Name of the private artifact which we will be referring to in our ARM template

Step 4: passing the parameter and referring to the artifact.

- Sample reference to the artifact will look like below:



The screenshot shows a code editor with the file path `master / ARMTemplates / deployVms.json`. The file name `deployVms.json` is highlighted. Below the file name, there are tabs for `Contents` and `Highlight changes`. A commit message `Committed d6035af6: Added install-package.ps1` is visible. The code content is as follows:

```
1 {  
2   "artifactId": "[resourceId('Microsoft.DevTestLab/labs/artifactSources/artifacts', Demo, 'name of custom artifact', 'Install-Package')]",  
3   "parameters": [  
4     {  
5       "name": "pm1",  
6       "value": "value for pm1"  
7     },  
8     {  
9       "name": "pm2",  
10      "value": "value for pm2"  
11    },  
12    {  
13      "name": "pm3",  
14      "value": "value for pm2"  
15    },  
16    {  
17      "name": "PAT",  
18      "value": "PAT token value"  
19    }  
20  ]  
21 }
```

Annotations with red boxes and lines pointing to the code:

- Lab Name**: Points to the `Demo` value in the `artifactId` string.
- Name of the custom artifact that we got from step 3**: Points to the `'name of custom artifact'` string in the `artifactId` string.
- folder name in which our .ps1 and artifact json file is present**: Points to the `'Install-Package'` string in the `artifactId` string.

