

# Virtual Machine Scale Set Deployment

Via Python SDK





# Creating github repository

<https://github.com/JanB1989/AzureVmScaleset>

The screenshot shows the GitHub interface for the repository 'AzureVmScaleset' by user 'JanB1989'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows 2 commits by JanB1989, with the latest commit 'so far so good' (3f3c20b) from yesterday. The file list includes 'infra', '.gitignore', 'README.md', and 'deploy.py', all committed yesterday.

File	Commit Message	Commit Date
infra	so far so good	yesterday
.gitignore	Initial commit	yesterday
README.md	so far so good	yesterday
deploy.py	so far so good	yesterday



# Creating local Python virtual Environment (using venv) with necessary dependencies

`venv` in Python is a tool to create **virtual environments** — isolated spaces where you can install packages without affecting your system-wide Python or other projects.

Requirements.txt file for dependencies

≡ requirements.txt

```
1 azure-identity>=1.16.0,<2.0.0
2 azure-mgmt-resource>=23.0.0,<24.0.0
3 python-dotenv>=1.0.0,<2.0.0
4 azure-mgmt-compute>=33.0.0,<34.0.0
5
6 | Ctrl+L to chat, Ctrl+K to generate
```



# Creating infrastructure definitions

- Subscription level resource template (subscription.json)
- Resource group level template (main.json)
- Separate file for configuration parameters (dev.json)



# Create json templates for subscription level scope

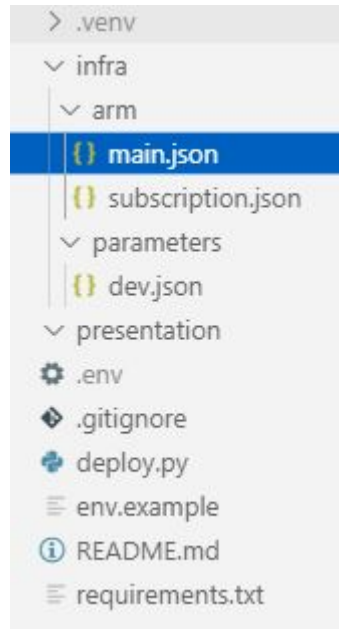
- Runs at subscription level, creates the resource group, then performs a nested deployment into that RG

```
> .venv
v infra
  v arm
    {} main.json
    {} subscription.json
  v parameters
    {} dev.json
  v presentation
  .env
  .gitignore
  deploy.py
  env.example
  README.md
  requirements.txt
```



# Create json template for resource group level scope

- Deploys resources into an existing resource group (in this case our created group).



# Create Configuration file with parameters for deployment

```
▼ AZUREVMSCALESSET
  > .venv
  ▼ infra
    ▼ arm
      {} main.json
      {} subscription.json
    ▼ parameters
      {} dev.json
  ▼ presentation
  ⚙ .env
  💎 .gitignore
  🍷 deploy.py
  ≡ env.example
  ⓘ README.md
  ≡ requirements.txt
```

```
infra > parameters > {} dev.json > ...
```

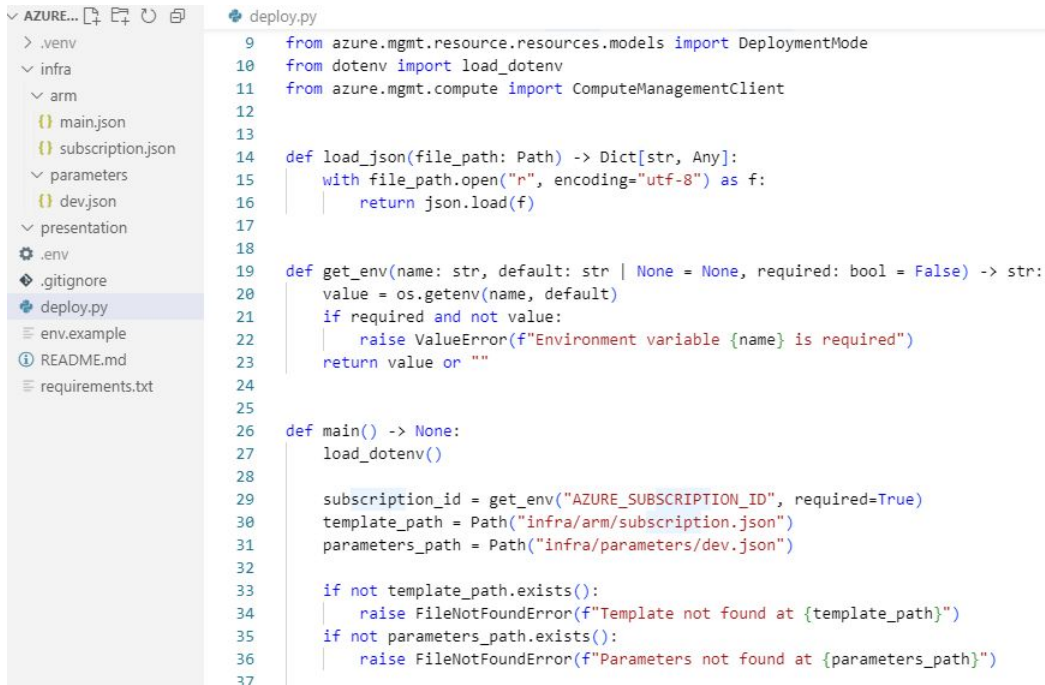
```
1  {
2    "rgName": "rg-vmss-demo",
3    "rgLocation": "germanywestcentral",
4    "vmSize": "Standard_B2s",
5    "instanceCount": 1,
6    "autoscaleEnabled": true,
7    "autoscaleMin": 1,
8    "autoscaleMax": 3,
9    "autoscaleDefault": 1,
10   "autoscaleMetricName": "Percentage CPU",
11   "autoscaleTimeWindow": "PT5M",
12   "autoscaleTimeAggregation": "Average",
13   "autoscaleStatistic": "Average",
14   "autoscaleOperatorScaleOut": "GreaterThan",
15   "autoscaleOperatorScaleIn": "LessThan",
16   "autoscaleThresholdScaleOut": 70,
17   "autoscaleThresholdScaleIn": 30,
18   "autoscaleChangeCountScaleOut": "1",
19   "autoscaleChangeCountScaleIn": "1",
20   "autoscaleCooldownScaleOut": "PT2M",
21   "autoscaleCooldownScaleIn": "PT2M",
22   "adminPublicKey": "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIB6UngofU2yx2twHT30FZESs8QSmYuWep5rT/02TLqCU vmss-demo
23 }
24
```



**Create Python deployment file to use arm templates and python azure SDK to deploy**



# Create Python deployment file to use arm templates and python azure SDK to deploy



```
9 from azure.mgmt.resource.resources.models import DeploymentMode
10 from dotenv import load_dotenv
11 from azure.mgmt.compute import ComputeManagementClient
12
13
14 def load_json(file_path: Path) -> Dict[str, Any]:
15     with file_path.open("r", encoding="utf-8") as f:
16         return json.load(f)
17
18
19 def get_env(name: str, default: str | None = None, required: bool = False) -> str:
20     value = os.getenv(name, default)
21     if required and not value:
22         raise ValueError(f"Environment variable {name} is required")
23     return value or ""
24
25
26 def main() -> None:
27     load_dotenv()
28
29     subscription_id = get_env("AZURE_SUBSCRIPTION_ID", required=True)
30     template_path = Path("infra/arm/subscription.json")
31     parameters_path = Path("infra/parameters/dev.json")
32
33     if not template_path.exists():
34         raise FileNotFoundError(f"Template not found at {template_path}")
35     if not parameters_path.exists():
36         raise FileNotFoundError(f"Parameters not found at {parameters_path}")
37
```



# Deploy infrastructure to Azure

```
● PS C:\Development\AzureVmScaleset> .venv\Scripts\Activate.ps1
● (.venv) PS C:\Development\AzureVmScaleset> python deploy.py
{
  "resourceGroupName": {
    "type": "String",
    "value": "rg-vmss-demo"
  },
  "vmssName": {
    "type": "String",
    "value": "vmss-demo"
  }
}
```

○ (.venv) PS C:\Development\AzureVmScaleset> █

2023/08/01



# What is actually deployed?

## Infrastructure

- Resource group: rg-vmss-demo (germanywestcentral) — created by subscription deployment
- Virtual network: vnet-demo (10.0.0.0/16)
- Subnet: subnet1 (10.0.0.0/24)
- VM scale set: vmss-demo — Ubuntu 22.04 LTS Gen2; size Standard\_B2s; instances 1; upgrade policy Manual; overprovision true
- Access: user azureuser; SSH only (key from dev.json); password auth disabled
- Networking: single NIC on vnet-demo/subnet1
- Bootstrap: cloud-init installs stress-ng (for running test script)



# What is actually deployed?

## Scaling rules and VMSS script extension

- Autoscale: enabled (true); capacity min/max/default = 1 / 3 / 1
- Scale out: metric Percentage CPU; window PT5M; aggregation Average; statistic Average; operator GreaterThan; threshold 70; change +1; cooldown PT2M
- Scale in: operator LessThan; threshold 30; change -1; cooldown PT2M
- Custom scripts: add VMSS Custom Script Extension to fetch/run bootstrap scripts alongside cloud-init (e.g., app install, config)



# Testing Autoscale

- Each VMSS instance is pre-provisioned with the `stress-ng` tool via cloud-init.
- We remotely start a short CPU load on all instances using Azure Run Command (no redeploy or SSH needed).
- The load runs for 5min (parameter) minutes, driving up **ALL** deployed VM CPU usage



# Triggering CPU load script on deployed VM's

- Set names and gather instance IDs:

```
$rg = 'rg-vmss-demo'; $vmss = 'vmss-demo'
$ids = az vmss list-instances -g $rg -n $vmss --query "[].instanceId" -o tsv
```

- Start load for 5 minutes (300s) on each instance:

```
foreach ($id in $ids) {
  az vmss run-command invoke -g $rg -n $vmss --instance-id $id `
    --command-id RunShellScript `
    --scripts "nohup stress-ng --cpu 0 --timeout 300s --metrics-brief >/tmp/stress.log 2>&1 &"
}
```



## **Check Result in Azure**