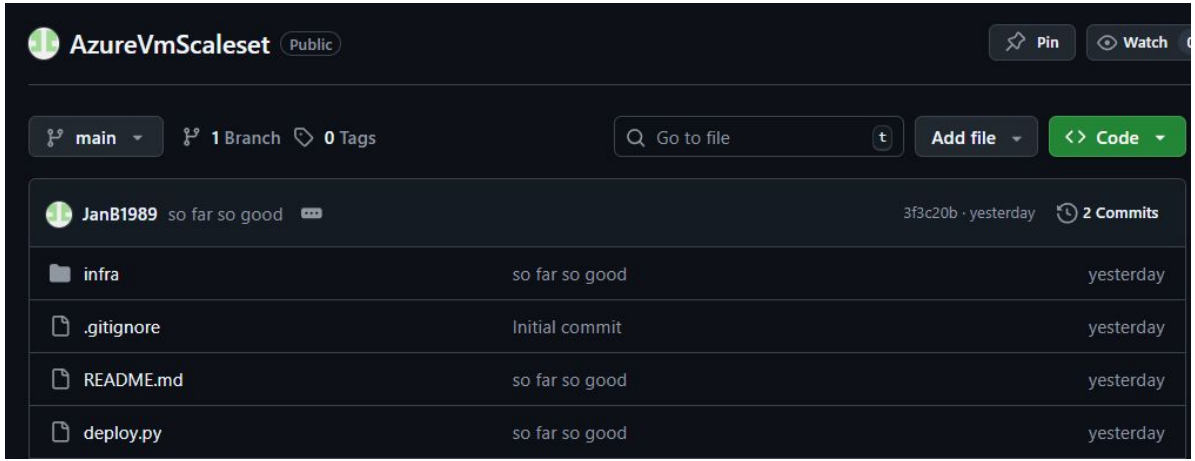
Virtual Machine Scale Set Deployment

Via Python SDK

Creating github repository

https://github.com/JanB1989/AzureVmScaleset



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

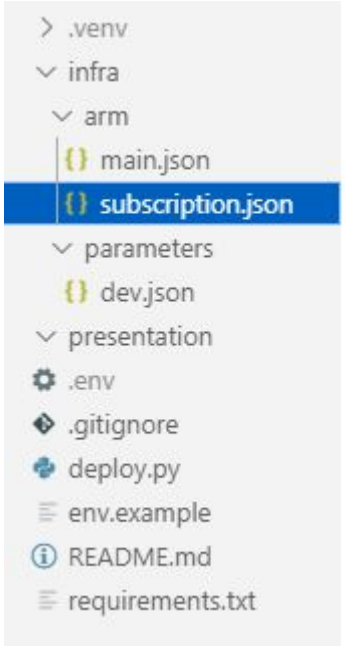
```
azure-identity>=1.16.0,<2.0.0
azure-mgmt-resource>=23.0.0,<24.0.0
python-dotenv>=1.0.0,<2.0.0
azure-mgmt-compute>=33.0.0,<34.0.0

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)

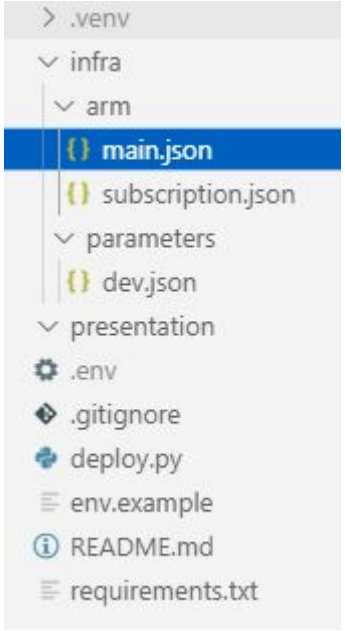
Create json templates for subscription level scope

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

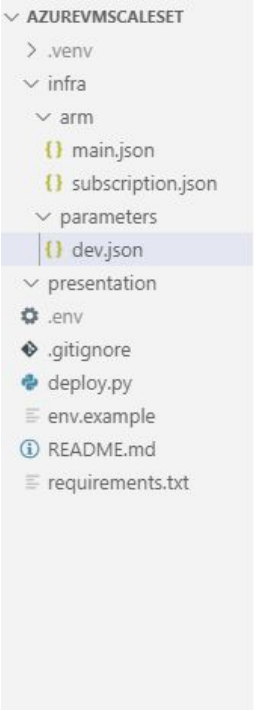


Create json template for resource group level scope

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



Create Configuration file with parameters for deployment

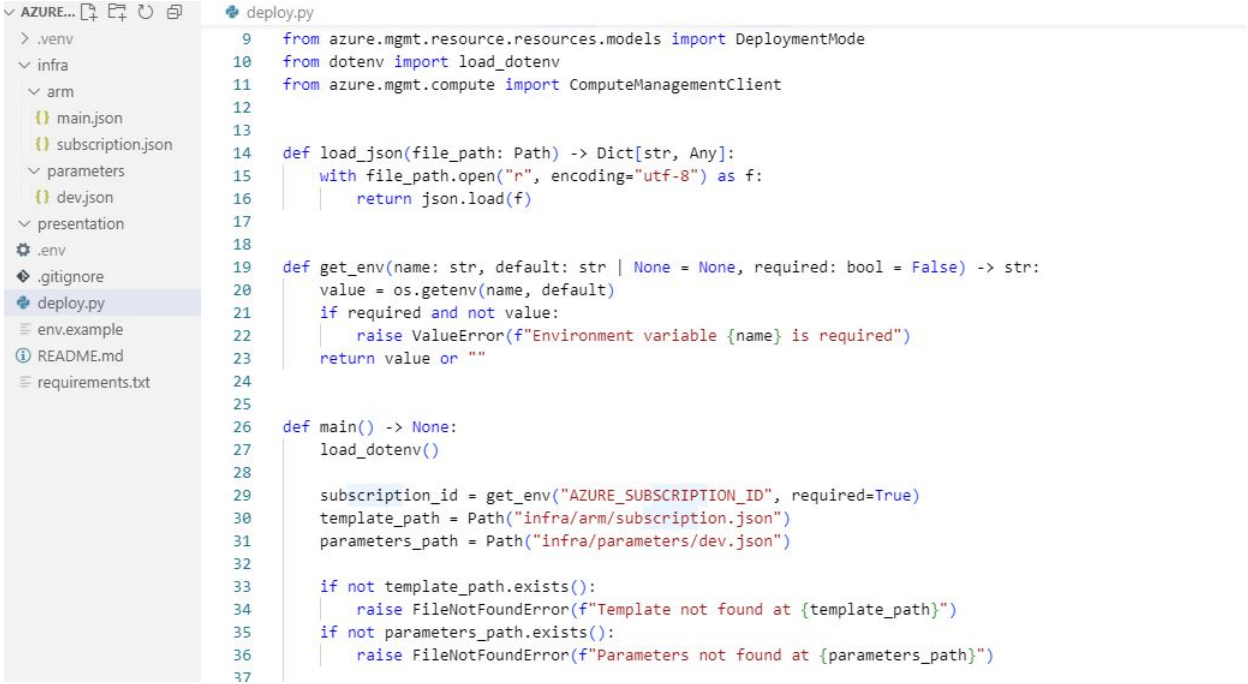


infra > parameters > {} dev.json > ...

```
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 AAAAC3NzaC1lZDI1NTE5AAAAIB6UrgofU2yx2tWHT3OFZESs8QSmyuWep5rT/02TLqcU vmss-demo
23  }
```

Create Python deployment file (that uses our templates)

Create Python deployment file (that uses our templates)




```python
from azure.mgmt.resource.resources.models import DeploymentMode
from dotenv import load_dotenv
from azure.mgmt.compute import ComputeManagementClient


def load_json(file_path: Path) -> Dict[str, Any]:
    with file_path.open("r", encoding="utf-8") as f:
        return json.load(f)


def get_env(name: str, default: str | None = None, required: bool = False) -> str:
    value = os.getenv(name, default)
    if required and not value:
        raise ValueError(f"Environment variable {name} is required")
    return value or ""


def main() -> None:
    load_dotenv()

    subscription_id = get_env("AZURE_SUBSCRIPTION_ID", required=True)
    template_path = Path("infra/arm/subscription.json")
    parameters_path = Path("infra/parameters/dev.json")

    if not template_path.exists():
        raise FileNotFoundError(f"Template not found at {template_path}")
    if not parameters_path.exists():
        raise FileNotFoundError(f"Parameters not found at {parameters_path}")
```

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>
```

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 Portal