# Scenario2

```
{
    "variables": {
        "aws_access_key": "XXX",
        "aws_secret_key": "XXX"
    },
    "builders": [
        {
            "type": "amazon-ebs",
            "access_key": "{{user `aws_access_key`}}",
            "secret_key": "{{user `aws_secret_key`}}",
            "region": "us-east-2",
            "source_ami_filter": {
                "filters": {
                    "virtualization-type": "hvm",
                    "name": "ubuntu/images/*ubuntu-xenial-16.04-amd64-server-*",
                    "root-device-type": "ebs"
                },
                "owners": [
                    "099720109477"
                ],
                "most_recent": true
            },
            "instance_type": "t2.micro",
            "ssh_username": "ubuntu",
            "ami_name": "scenario1-packer-{{timestamp}}"
        }
    ],
    "provisioners": [
        {
        "type": "file",
        "source": "./hello.txt",
        "destination": "/home/ubuntu"
        },
        {
            "type": "shell",
            "inline": [
                "ls -al /home/ubuntu",
                "cat /home/ubuntu/hello.txt"
            ]
        },
        {
            "type": "shell",
            "script":
                "./install_nginx.sh"
        }
```

```
    ]
}
```

## Variables

```
jsonCopy code
"variables": {
    "aws_access_key": "XXX",
    "aws_secret_key": "XXX"
}
```

Les clés d'accès AWS sont définies ici, bien que, pour des raisons de sécurité, il soit recommandé de les passer comme variables d'environnement ou d'utiliser un système de gestion des secrets.

## Builders

Le "builder" est du type `amazon-ebs`, ce qui signifie qu'il créera une image Amazon Elastic Block Store (EBS).

- `access_key` **et** `secret_key`**:** Utilisés pour l'authentification auprès d'AWS.
- `region`**:** La région AWS où l'image sera créée.
- `source_ami_filter`**:** Filtre pour sélectionner l'AMI source.
  - `filters`**:** Critères pour le filtrage (type de virtualisation, nom, type de périphérique racine).
  - `owners`**:** ID du propriétaire de l'AMI source.
  - `most_recent`**:** Utiliser l'AMI la plus récente qui correspond aux filtres.
- `instance_type`**:** Type d'instance EC2 à utiliser pour la création.
- `ssh_username`**:** Utilisateur SSH pour se connecter à l'instance.
- `ami_name`**:** Nom de la nouvelle AMI, avec un timestamp pour l'unicité.

## Provisioners

Ce sont des étapes pour installer et configurer le logiciel sur l'instance avant de créer l'AMI.

1. **Provisioner de type "file":**

- `source` **:** Chemin du fichier sur votre machine locale ( `./hello.txt` ).
- `destination` **:** Chemin où le fichier doit être copié sur l'instance ( `/home/ubuntu` ).

1. **Premier provisioner de type "shell":**

- `inline` **:** Commandes shell à exécuter sur l'instance.
  - `ls -al /home/ubuntu` : Liste les fichiers dans le répertoire `/home/ubuntu` .
  - `cat /home/ubuntu/hello.txt` : Affiche le contenu du fichier `hello.txt` .

1. **Deuxième provisioner de type "shell":**

- `script` **:** Chemin vers un script shell à exécuter sur l'instance ( `./install_nginx.sh` ).

Le nouveau provisioner du type "shell" permet d'exécuter un script shell ( `install_nginx.sh` ) qui est présumé être sur votre machine locale. Ce script pourrait installer Nginx ou faire toute autre configuration nécessaire sur l'instance.

Le fichier `scenario2.json` est donc une configuration complète pour créer une image Amazon AMI personnalisée, y compris le transfert de fichiers et l'exécution de commandes shell et de scripts sur l'instance avant la création de l'AMI.