Preparing for the Cluster Setup: Availability Zones and SSH Keys

In this lesson, we will set up availability zones and create SSH keys.

Setting Up the Availability Zones

In this lesson, we will decide which availability zones should we use. So, let's take a look at what's available in the `us-east-2` region.

```
aws ec2 describe-availability-zones \
    --region $AWS_DEFAULT_REGION
```

The **output** is as follows.

```
{
    "AvailabilityZones": [
        {
            "State": "available",
            "RegionName": "us-east-2",
            "Messages": [],
            "ZoneID": "use2-az1",
            "ZoneName": "us-east-2a"
        },
        {
            "State": "available",
            "RegionName": "us-east-2",
            "Messages": [],
            "ZoneID": "use2-az2",
            "ZoneName": "us-east-2b"
        },
        {
            "State": "available",
            "RegionName": "us-east-2",
```

```
            "RegionName": "us-east-2",
            "Messages": [],
            "ZoneID": "use2-az3",
            "ZoneName": "us-east-2c"
        }
    ]
}
```

As we can see, the region has three availability zones. We'll store them in an environment variable.

📝 A note to Windows users

> Please use `tr '\r\n' ', '` instead of `tr '\n' ','` in the command that follows.

```
export ZONES=$(aws ec2 \
    describe-availability-zones \
    --region $AWS_DEFAULT_REGION \
    | jq -r \
    '.AvailabilityZones[].ZoneName' \
    | tr '\n' ',' | tr -d ' ')

ZONES=${ZONES%?}

echo $ZONES
```

Just as with the access keys, we used `jq` to limit the results only to the zone names, and we combined that with `tr` that replaced new lines with commas. The second command removes the trailing comma.

The output of the last command that echoed the values of the environment variable is as follows.

```
us-east-2a,us-east-2b,us-east-2c
```

We'll discuss the reasons behind the usage of three availability zones later on. For now, just remember that they are stored in the environment variable `ZONES`.

🔑 Creating SSH Keys

The last preparation step is to create SSH keys required for the setup. Since we might create some other artifacts during the process, we'll create a directory

dedicated to the creation of the cluster.

```
mkdir -p cluster

cd cluster
```

SSH keys can be created through the `aws ec2` command `create-key-pair`.

```
aws ec2 create-key-pair \
    --key-name devops23 \
    | jq -r '.KeyMaterial' \
    >devops23.pem
```

We created a new key pair, filtered the output so that only the `KeyMaterial` is returned, and stored it in the `devops23.pem` file.

For security reasons, we should change the permissions of the `devops23.pem` file so that only the current user can read it.

```
chmod 400 devops23.pem
```

Finally, we'll need only the public segment of the newly generated SSH key, so we'll use `ssh-keygen` to extract it.

```
ssh-keygen -y -f devops23.pem \
    >devops23.pub
```

All those steps might look a bit daunting if this is your first contact with AWS. Nevertheless, they are pretty standard. No matter what you do in AWS, you'd need to perform, more or less, the same actions. Not all of them are mandatory, but they are good practices. Having a dedicated (non-admin) user and a group with only required policies is always a good idea. Access keys are necessary for any `aws` command. Without SSH keys, no one can interactively log in to a server.

The good news is that we're finished with the prerequisites. In the next lesson, we can turn our attention towards creating a Kubernetes cluster.