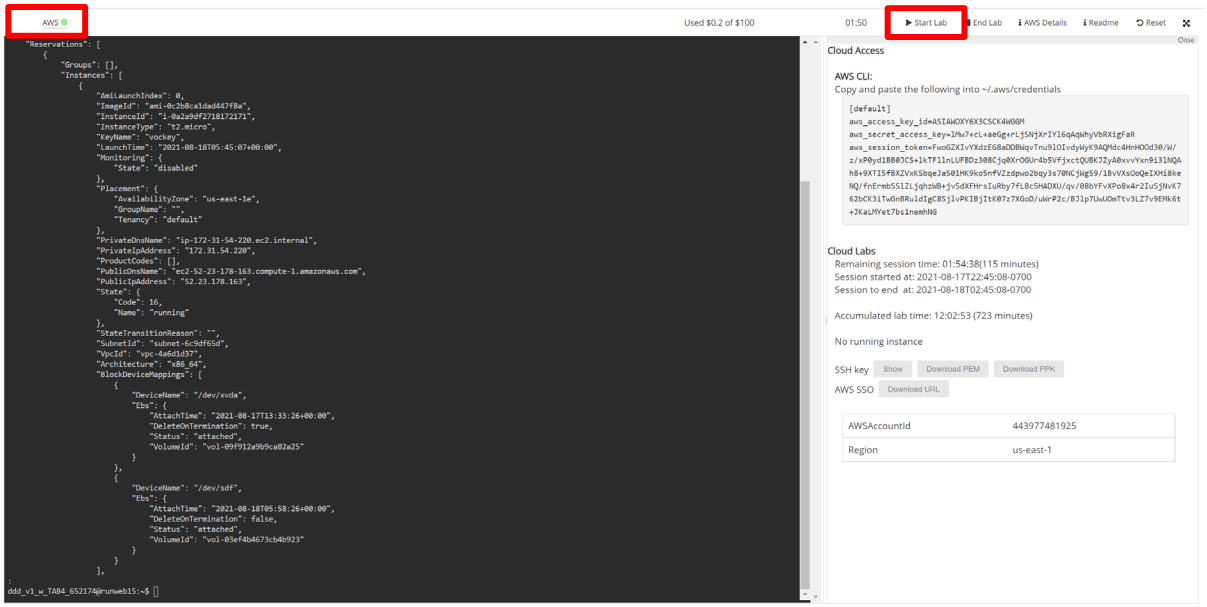
Lab 3 – Amazon EC2

In this lab, we will create and work with Amazon EC2 instances. Document all your steps including screenshots if necessary. After the lab, you will have to create a lab report, where you summarize your activities.

1 Amazon Web Service Portal

1.1 Preparations

We will access AWS through the AWS Academy. Log into the Academy portal, select “Module” and then “Learner Lab”. There you can start your lab, cf. screenshot below.



To start your session, click “Start Lab” on the top right side of the page. As soon as the dot on the top left of the page turns green, you can click the AWS link next to it to access the AWS portal.

Task: Login

- Login to your Amazon Web Services Portal using AWS Academy
- **If not specified otherwise, all steps in this section are performed in the AWS portal.**

1.2 Launch of an EC2 Instance

Task: Launch an instance in EC2

- Determine the AWS region in which you want to launch the Amazon EC2 Instance
  - Must be `us-east-1` for the education accounts
- Launch an Amazon EC2 instance from a preconfigured Amazon Machine Image (AMI)
  - Use “Amazon Linux 2”
- Choose an instance type
  - Free Tier
- Configure network, IP address, security groups and tags
  - Default configuration is fine

- Make sure to select or create a key pair if prompted
  - The "vockey" key can be used using the pem/ppk files from AWS Academy
  - Optional: Convert the key for use with putty (puttygen)
  - Optional: You might have to change the Linux permissions for your key file. If get the message "WARNING: UNPROTECTED PRIVATE KEY FILE!" when attempting to connecting to your EC2-VM using ssh, execute the following command:
    `chmod 600 <private-key-file>`

1.3 Security Groups

Task: Experiment with Security Groups

- Make sure your security group is configured to allow SSH connections
- Try to login to your instance using SSH (e.g., putty, ssh)
  - Identify the public IP address using the Portal
  - User: `ec2-user`
  - On Linux: `ssh -i <private-key-file> <user>@<ip>`
- Setup Webserver (on EC2-VM!)
  - Use `sudo` to run the commands as root
  - Install Apache Server:
    - `yum install httpd -y`
  - Turn on the Server:
    - `service httpd status` → server stopped
    - `service httpd start` → start server
    - `chkconfig httpd on` → enable service
    - Check if server is running (service httpd status)
  - Go to root directory of the web server:
    - `/var/www/html`
  - Create a html page using vi or nano
    - `index.html`
- Try to access Webserver
  - Open `<IP>/index.html` in browser
  - → Error
- Allow access
  - Modify Security Group to enable access to the Webserver (port 80)
- Experiment with Security Group
  - Allow Access to Webserver in different ways
    - From your exact IP only
    - From all IPs
    - From all IPs in your subnet
    - By selecting a port range
  - Create a separate Security Group
    - Allow HTTP access for everyone
    - Attach group to instance
  - Final Setup
    - Initial security group allows SSH access
    - Additional security group allows HTTP access

1.4 Volumes and Snapshots

Volumes exist on EBS and can be seen as virtual hard disks. Snapshots of these volumes can be placed on S3 storage. In this part, we will create a volume and attach it to an instance. Then we will create a snapshot from the volume and use that snapshot to create another volume.

Task: Experiment with Volumes and Snapshots

- Create another volume and attach it to your instance
  - 1 GB is enough
  - Important: same availability zone as instance
- Check if your instance has the new volume
  - List block devices: `lsblk`
  - Check type: `file -s /dev/xvdf`
- Create filesystem and mount
  - Create ext4 partition: `mkfs -t ext4 /dev/xvdf`
  - Create directory to mount: e.g., `mkdir /fileserver`
  - Mount partition to mount point: `mount /dev/xvdf /fileserver`
- Create file on new partition and unmount
  - E.g., `touch i-was-here`
  - Unmount: `umount /dev/xvdf`
- Detach volume from instance
  - List block devices to see if volume detached
- Create Snapshot of Volume
- Recreate Volume from Snapshot
  - Delete Volume
  - Create Volume from Snapshot
  - Attach new volume to instance
  - Mount volume in instance
  - Check if file “i-was-here” exists

1.5 Load Balancers

Task: Experiment with Load Balancers

- Launch a second instance
  - Same settings as first instance, except: use different availability zone
    - To launch in a different availability zone, chose a subnet from that zone
    - Use ONLY the security group for SSH created for first instance
  - Install Webserver
  - Create index.html with different content (e.g., “Hello World from instance 2!”)
- Create Load Balancer: Application Load Balancer
  - Choose name
  - Scheme: internet-facing to allow internet access
  - IP address type: IPv4
  - Listener: Listen to HTTP traffic on port 80
  - Availability Zones: Choose availability zones used for the two instances
  - Security Groups: Attach the http security group
  - Routing: Create new Target Group
    - Choose name
    - Target type: instance (select instances as targets, rather than IPs)

        - Protocol/Port: HTTP/80
        - Health check: default values
    - Targets: Register the two instances as targets
    - Create and wait for provisioning
- Find and copy Load Balancer DNS name
    - Access it through Web browser → reach instance 1
- Check Health status of instances in the created target group
    - instance 2 is unhealthy → why?
    - Adapt configuration of instance 2 to become healthy
- Access Load Balancer multiple times (reload)
    - What happens?

1.6 Path-based Load Balancing

In this part of the lab, we will create a path-based rule for the Load Balancer to forward path `/i1` to the first instance and path `/i2` to the second instance.

Note: Path-based rules cannot be created upon initial creation of the load balancer, but must be added afterwards.

Note: The directories `/i1` and `/i2` respectively, must be created on the respective instances and must contain an `index.html` file.

Task: Experiment with Path-based Load Balancing

- Use the provided PDF containing a blog post by Zeeshan Baig to set up the past-based load balancing as required
    - Source (no longer available): https://medium.com/@zeebaig/path-based-routing-in-aws-application-load-balancer-b5a91a79d7f9

2 AWS CLI

In this part of the lab we will use the Command Line Interface to interact with AWS.

2.1 Setup

There are 4 ways to use the AWS CLI for the following part of the lab:

1. The provided VM has the AWS CLI pre-installed.
2. Download and install the AWS Command Line Interface from: https://aws.amazon.com/cli
3. If you do not want to install the CLI locally, and also do not want to use the VM, you can use the CLI remotely on an EC2 instance. Since the CLI is installed on EC2 instances by default, you can use SSH/Putty to connect to an instance of your choice and run the CLI there (cf. “Remotely” below.)
4. The AWS Academy website includes a console which is preconfigured to allow interaction with AWS. If you use the included console, you can skip the configuration part (2.3) below, as it is already preconfigured. However, I do not recommend using this console as it is rather slow.

2.2 About the CLI

Taken from https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html

The AWS Command Line Interface (AWS CLI) is an open source tool that enables you to interact with AWS services using commands in your command-line shell. With minimal configuration, the AWS CLI enables you to start running commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from the command prompt in your terminal program:

- **Linux shells** – Use common shell programs such as bash, zsh, and tcsh to run commands in Linux or macOS.
- **Windows command line** – On Windows, run commands at the Windows command prompt or in PowerShell.
- **Remotely** – Run commands on Amazon Elastic Compute Cloud (Amazon EC2) instances through a remote terminal program such as PuTTY or SSH, or with AWS Systems Manager.

All IaaS (infrastructure as a service) AWS administration, management, and access functions in the AWS Management Console are available in the AWS API and CLI.

For further information about the CLI, see:

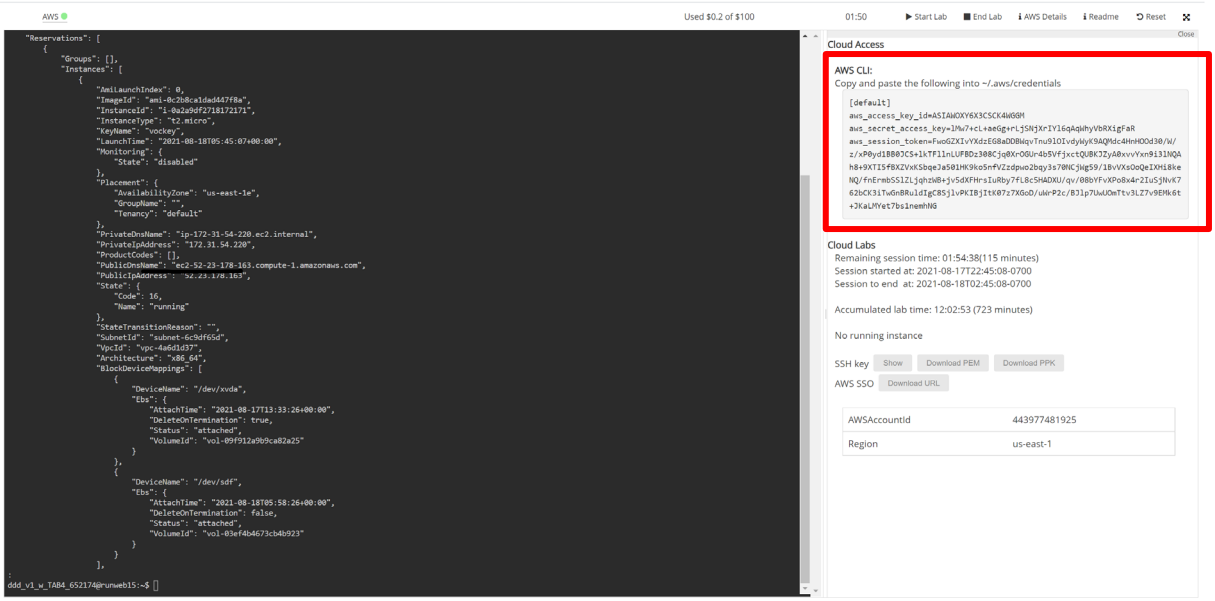
- Using the AWS CLI: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-using.html
- Command reference: https://awscli.amazonaws.com/v2/documentation/api/latest/reference/index.html

2.3 Configuration

Please read section 2.3 entirely before you start your configuration!

In general, the command `aws configure` configures the CLI installation. It will prompt for Access Key data, AWS region and output format, and create a profile "default" if not instructed otherwise.

- For access key and secret access key use the data provided in the AWS Academy portal → AWS CLI section (see screenshot below)
- As region you MUST use us-east-1
- Output format should be json



Further information on configuration can be found here:
https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html

The credentials are stored in the user home directory in `~/.aws`, where ~ represents the users home folder. The file `credentials` contains the profile information, e.g.:

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Add the `aws_session_token` from the AWS Academy web page to this file. Alternatively, you can just copy the credentials from the AWS Academy into the file. Note however, that the `~/.aws` directory might not yet exists and the file `config` (cf. below) will not be created automatically. Directory and files an however be created manually, if preferred.

The file `config` contains additional configuration, like region and output format, e.g.:

```
[default]
region=us-east-1
output=json
```

Task: Configure your CLI

- Either
  - Use `aws configure` and copy the *session token*, or
  - Create and configure the two files manually
- Check the content of the two files
- Run the command `aws ec2 describe-instances`
  - If it returns an error, something went wrong
  - Otherwise your CLI is successfully configured

2.4 AWS Command Structure

Taken from https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-commandstructure.html

The AWS CLI uses a multipart structure on the command line that must be specified in this order:

- The base call to the **aws** program.
- The **top-level command**, which typically corresponds to an AWS service supported by the AWS CLI.
- The **subcommand** that specifies which operation to perform.
- General CLI **options or parameters** required by the operation. You can specify these in any order as long as they follow the first three parts. If an exclusive parameter is specified multiple times, only the last value applies.
  Parameters can take various types of input values, such as numbers, strings, lists, maps, and JSON structures. What is supported is dependent upon the command and subcommand you specify.

```
aws <command> <subcommand> [options and parameters]
```

Additionally, some AWS services have wait commands available. Any command that uses `aws wait` usually waits until a command is complete before it moves on to the next step.

2.5 AWS CLI WITH EC2

2.5.1 Key-Pairs

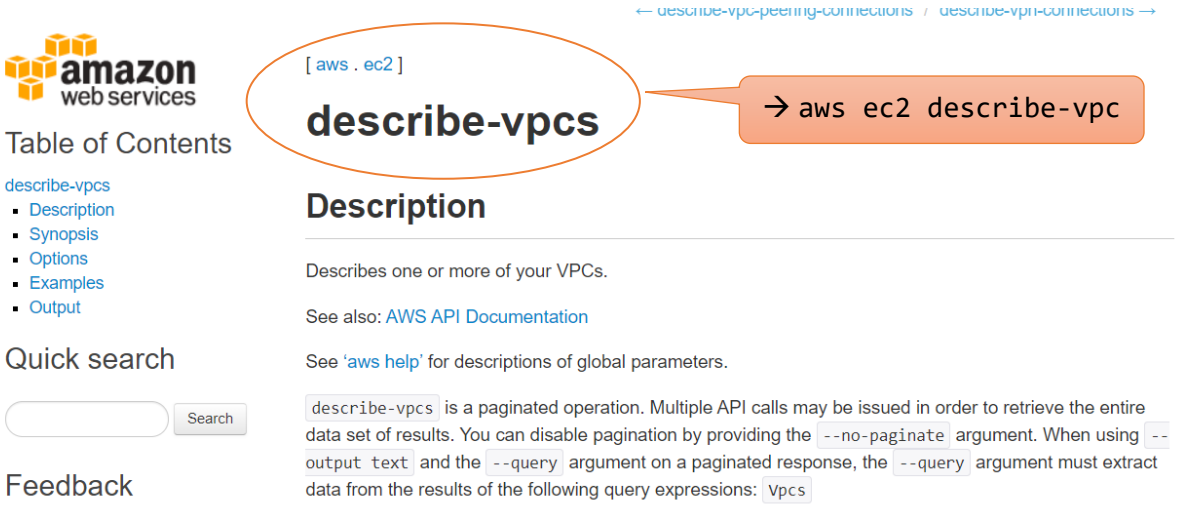
In the first part of the lab, you created Key-Pair using the AWS service portal. This key can be displayed using the command:

```
aws ec2 describe-key-pairs
```

2.5.2 Virtual Private Cloud – VPC

New EC2 instances are usually launched into a subnet of a VPC. To do so, we have to identify the ID of the VPC we used in the previous section (the default VPC). To find out, how to identify, we can consult the CLI command reference (link see above).

In the "available services" section of the overview, we select "EC2", since we want to work with instances. The available commands for ec2 are displayed. The commands used to list information are all prefixed with "describe-…". Thus, since we want to list all VPCs we choose the describe-vpcs command. The page shows how to use the selected command. On the top of the page, we see the command itself, and below the options are explained.



The output includes the VpcId. Using this ID we can identify the subnets associated with this VPC, using the respective describe command for subnets.

- Use the CLI to list the subnets
  - Filter by the VpcId of the default subnet
  - See the Command reference for details on filtering

2.5.3 Run instance

To run an instance, we should at least have the following information (even though there are default values for most of the parameters):

- Subnet
- Security Group
- Instance Type (Helpful filter available!)
- Key Name
- Image Id

Tasks

- List all subnets, security groups, keys and instance types using the CLI
  - Use filters where appropriate
  - For the Image Id, find and use the Id of an Amazon Linux 2 image
- Extract the data needed for the launch of your instance

Run an instance into the same **security group** (for SSH) and **subnet** as in the first part, using the identified **key-name**, the provided **image id** and an **instance type** from the free tier.

2.5.4 Querying output

While the instance is launching, query the state of your instances using the `describe-instances` command. The output includes the `InstanceIds` and `State` of all instances, but also lots of currently unrelated information. Use the --query parameter to create a Query on the output of the command, that produces an output like below:

```
[
    [
        {
            "Id": "i-08e1e1e5dd720cef0",
            "State": "running"
        }
    ]
]
```

This output includes only Instance Ids and their current state.

Hint: The following User Guide might be helpful:
https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-output.html

2.6 Working with the CLI

Repeat the steps you performed using the AWS portal with the CLI. Do not rely on default values, but specify parameters as needed (e.g., VPC id, …). Include all necessary queries in your lab report.

Hint: The command reference might be helpful:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/index.html

- Create a second instance
  - Monitor instance status with your query
- Security Groups
  - Create a new security group, allowing HTTP traffic from your IP only
  - Add new HTTP Security Group to one instance
- Create Volumes and Snapshots
- Experiment with Load Balancer
  - Create new Load Balancer (and new Target Group)
  - Remove and add new HTTP Security Group to instances
  - Try to access your instances
  - Command: `aws elbv2 …`