AWS Academy Learner Lab [28589]

Practice 1

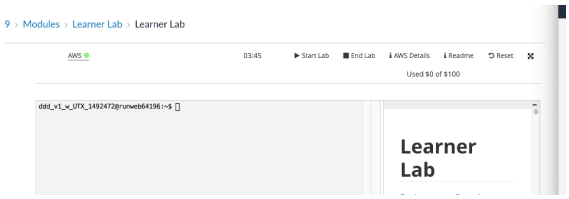
Introduction

Whenever we want to deploy resources in AWS, we need to consider which regions and availability zones we are going to use, as well as the restrictions we could have. We are being asked to complete these tasks:

- Use the Cloud Shell to create a bash/AWS CLI script that lists all AWS regions with their availability zones.
- Create another script that lists the maximum number of type T EC2 instances that we are allowed to create in Northen Virginia region (us-east-1)

Procedure

We click on Start Lab in order to do so, and we also want to open the AWS Management Console tab (even though we are not really using it in this learner lab) so we can take our required screenshots. An AWS CLI shell will open after some time.



We need to read the AWS CLI documentation to know how to list regions and availability zones:

https://docs.aws.amazon.com/cli/latest/reference/ec2/describe-availability-zones.html

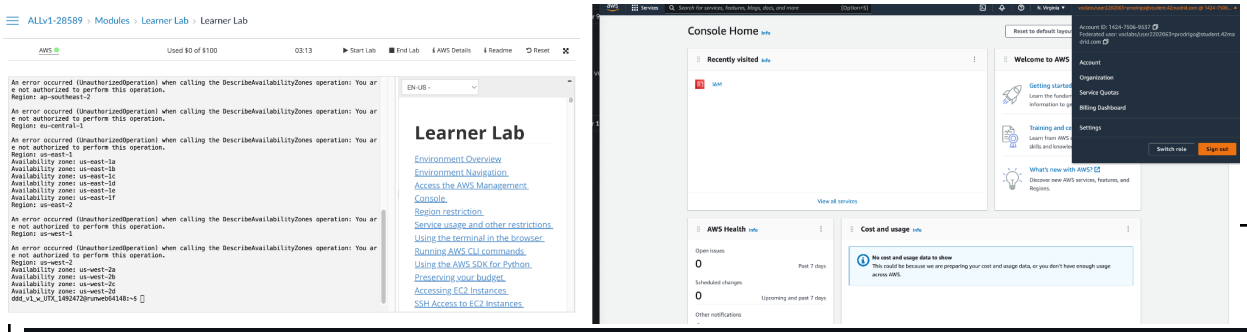
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-availability-zones

https://docs.aws.amazon.com/cli/latest/reference/ec2/describe-regions.html

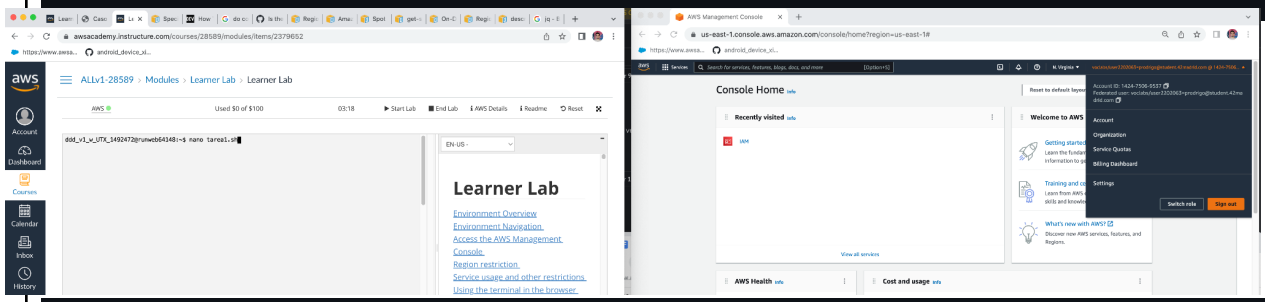
As our learner lab documentation says:

> **Region restriction**
>
> All service access is limited to the **us-east-1** and **us-west-2** Regions unless mentioned otherwise in the service details below. If you load a service console page in another AWS Region you will see access error messages.

We clearly are going to use only those regions, so we can either use describe-availability-zones with the region flag or we could create a script that gets all AWS regions (with opt-in filters or not) and then do describe-availability-zones on them. This will show multiple errors, due to the previously mentioned restrictions, but it will be more useful in a real case scenario. Once we created the script we can use either vim or nano to save it.




```
#!/bin/bash
```




```
# Task 1
REGIONS=$(aws ec2 describe-regions --all-regions | jq -r '.Regions | .[] |
.RegionName + " " + .OptInStatus'  | grep -v not-opted-in | cut -d' ' -f1)

for i in $REGIONS
   do
       echo -e "Region: $i"
       AVALZONE=$(aws ec2 describe-availability-zones --region $i --query
'AvailabilityZones[*].ZoneName' --output text)
       for j in $AVALZONE
           do
               echo -e "Availability zone: $j"
           done
   done

# Uncomment to get a more eye-pleasing version when you know which regions you are
using.
# aws ec2 describe-availability-zones --all-availability-zones --region us-east-1
--query 'AvailabilityZones[*].{RegionName:RegionName,ZoneName:ZoneName}' --output
table &&  aws ec2 describe-availability-zones --all-availability-zones --region
us-west-2 --query 'AvailabilityZones[*].{RegionName:RegionName,ZoneName:ZoneName}'
--output table
```

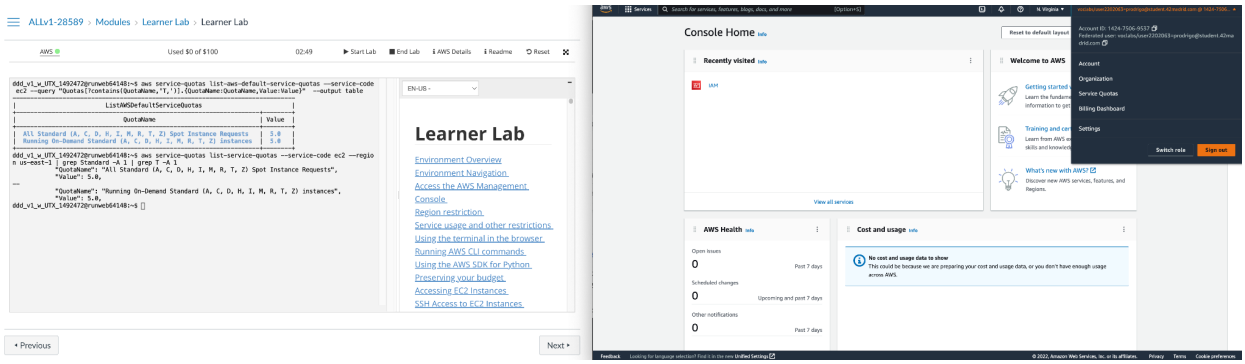
For our second task, we need to read the CLI service quota documentation:

https://docs.aws.amazon.com/cli/latest/reference/service-quotas/get-service-quota.html

We want to get the contents of QuotaName (as it tells us which type of server it is), and filter by ServiceCode EC2 (as we are being asked for EC2 instances) as well by the Northern Virginia region.

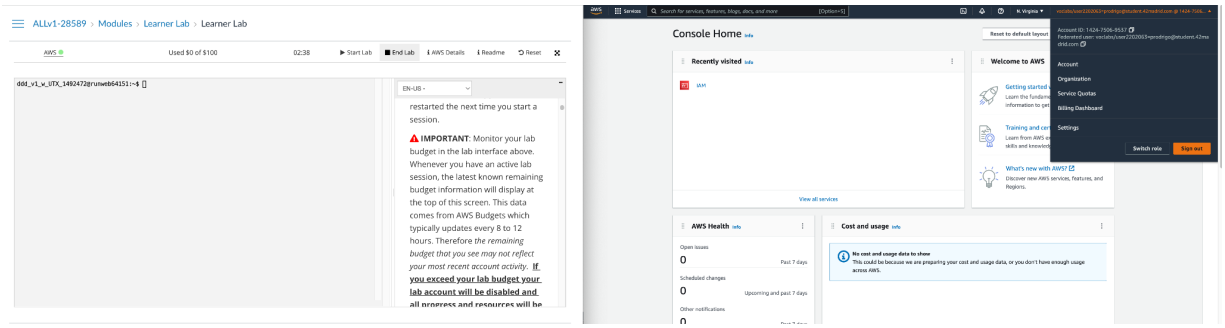
After filtering the data, either we use grep to get the next line which has the maximum number of instances (as “Value” field), and then we use grep again to filter only Value, or else we can get the data we need with a query.

```
#!/bin/bash

# Task 2

# First QuotaName and Value params in the brackets can be exchanged with

# both “Instance Type” and “Number of instances” respectively for better readability.

aws service-quotas list-aws-default-service-quotas --service-code ec2 --query
"Quotas[?contains(QuotaName,'T,')].{QuotaName:QuotaName,Value:Value}"  --output table

# or else, uncomment for a grep based command

# aws service-quotas list-service-quotas --service-code ec2 --region us-east-1 | grep
Standard -A 1 | grep T -A 1
```



We are able to create instances in the AWS Management Console to check that we really do have 5 maximum instances, but you don’t want to do that as you can get your credits drained pretty quickly.

Once we are done, we click on End Lab and Logout.



Closure

With automation tasks or working on JSON data in mind, AWS CLI is a very versatile tool. However, you might want to use a SDK or CloudFormation templates as an alternative, or to simplify the process.