# AWS CLI H4 – Networking

**Create a VPC with specs**

with following specs:

* Name: wordpress
* IPv4 CIDR block: 10.0.0.0/16
* No IPv6 CIDR block
* Tenancy: Default
* Number of Availability Zones: 1 (us-east-1a)
* Number of public subnets: 1
* Number of private subnets: 1
* NAT gateways: None
* VPC endpoints: None

*aws ec2 create-vpc --cidr-block 10.0.0.0/16 --instance-tenancy default --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=wordpress}]'*

Find VPC id with: *aws ec2 describe-vpcs*

*aws ec2 create-subnet --vpc-id <vpc\_id> --cidr-block 10.0.1.0/24 --availability-zone us-east-1a --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=public-subnet}]'*

*aws ec2 create-subnet --vpc-id <vpc\_id>--cidr-block 10.0.2.0/24 --availability-zone us-east-1a --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=private-subnet}]'*

**How to make the ‘public’ subnet public**

Create an Internet Gateway:

*aws ec2 create-internet-gateway*

Attach the Internet Gateway to the VPC:

*aws ec2 attach-internet-gateway --internet-gateway-id <internet\_gateway\_id> --vpc-id <vpc\_id>*

Create a new route table:

*aws ec2 create-route-table --vpc-id <vpc\_id> --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=public-routes}]'*

Add a route to the route table that points to the Internet Gateway:

*aws ec2 create-route --route-table-id <route\_table\_id> --destination-cidr-block 0.0.0.0/0 --gateway-id <internet\_gateway\_id>*

Associate the public subnet with the new route table:

*aws ec2 associate-route-table --route-table-id <route\_table\_id> --subnet-id <public\_subnet\_id>*

**Map public IP on launch**

*aws ec2 modify-subnet-attribute --subnet-id $(aws ec2 describe-subnets --filters Name=tag:Name,Values=public-subnet --query 'Subnets[0].SubnetId' --output text) --map-public-ip-on-launch*

**Create new Vockey and download**

*aws ec2 create-key-pair --key-name wordpress --query 'KeyMaterial' --output text > ~/.ssh/wordpress.pem****Note:***if wordpress key already exsist in AWS you will see an error message

list and delete key pairs:

*aws ec2 describe-key-pairs*

*aws ec2 delete-key-pair --key-name my-key-pair*

**Configuring the front-end application to use the VPC and key**

* AMI ID: the ID of the base docker compose image AMI from the compute lab.
* Security group ID: the ID of the wordpress security group that you created in this lab
* Keypair name: the keypair name that you've created in the previous step: wordpress
* Subnet ID: the ID of the public wordpress VPC subnet.
* API gateway URL: just use the value placeholder for now

describe commands:

*aws ec2 describe-images –owner self*

*aws ec2 describe-security-groups*

*aws ec2 describe-key-pairs*

*aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-12345678" --query "Subnets[\*].[SubnetId,CidrBlock]"*

**Modify IAM Profile for specific instance** (doesn’t always work if not admin)

*aws ec2 modify-instance-attribute --instance-id i-0760a0e8848d17cb9 --iam-instance-profile Name=LabInstanceProfile*

List possible IAM roles

*aws iam list-roles | grep ‘pattern’*

**Create Copy of Front-end in different Subnet**

Since the new Image has to be in a different subnet, first we need subnetid  
  
 *aws ec2 describe-subnets --filters "Name=vpc-id,Values=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query "Vpcs[0].VpcId" --output text)"*

find out which subnet previous instance is using, and SG

*aws ec2 describe-instances | grep ‘Value\|InstanceId\|Subnet\|GroupId’*

Create new instance with different subnet

*aws ec2 run-instances --image-id ami-082d43d4318a18c8b --instance-type t2.micro --key-name vockey --security-group-ids sg-0d15078df97c7b4f4 --subnet-id subnet-0c547963f0cc45143 --tag-specifications 'ResourceType=instance, Tags=[{Key=Name, Value=front-end2}]' --count 1 --placement AvailabilityZone=us-east-1e*

**Configure Elastic IP for the backend**

*aws ec2 allocate-address --domain vpc*

*aws ec2 associate-address --instance-id <instance-id> --allocation-id <allocation-id>*

to describe previously made addresses or association id’s

*aws ec2 describe-addresses*

**Create a Target Group**

with following specs

* Target group name: awsgen-frontend-tg
* Protocol: Http on port 80
* VPC: select the default VPC
* Protocol version: HTTP1
* Health checks: default settings

*aws elbv2 create-target-group --name awsgen-frontend-tg --protocol HTTP --port 80 --target-type instance --vpc-id <default-vpc-id> --protocol-version HTTP1*

Describe VPCid

*aws ec2 describe-vpcs | grep -i ‘Value\|vpc’*

List Taget groups and load balancers

*aws elbv2 describe-target-groups*

*aws elbv2 describe-load-balancers*

**Create Load Balancer for specific target group**

Specifications for LB:

* Load balancer name: awsgen-frontend-lb
* Scheme: internet-facing
* Ip address type: IPv4
* VPC: select the default VPC
* Mappings: select the 2 availability zones / subnets that are linked to the frontend and frontend-2 instances
* Security groups: awsgen app sg (that contains an allow rule on incomming http traffic)
* Listeners and routing: Add a listener on http (port 80) traffic and make sure ou forward the traffic to the awsgen-frontend-tg that we just created  
  *aws elbv2 create-load-balancer --name awsgen-frontend-lb --scheme internet-facing --type application --ip-address-type ipv4 --subnets subnet-06d3f0d345cfb2161 subnet-0c547963f0cc45143 --security-groups sg-0d15078df97c7b4f4 --tags Key=Name,Value=awsgen-frontend-lb*

*aws elbv2 create-listener --load-balancer-arn <your-load-balancer-arn> --protocol HTTP --port 80 --default-actions Type=forward,TargetGroupArn=<your-target-group-arn>*

*aws elbv2 create-listener --load-balancer-arn arn:aws:elasticloadbalancing:us-east-1:098739458718:loadbalancer/app/awsgen-frontend-lb/77066994eeea5fea --protocol HTTP --port 80 --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-east-1:098739458718:targetgroup/awsgen-frontend-tg/f94c76b77fd2edb7*

**Create a new launch template**

Specifications

Name: awsgen-frontend-launch-template

Make sure Auto Scaling guidance is checked

AMI: awsgen-frontend

Instance type: t2.micro

Key pair: vockey

Subnet: Don't include in launch template

Security group: awsgen app sg

*aws ec2 create-launch-template --launch-template-name awsgen-frontend-launch-template --version-description "Version 1" --launch-template-data "ImageId=ami-082d43d4318a18c8b,InstanceType=t2.micro,KeyName=vockey,SecurityGroupIds=sg-05962f1788f5be339”*

describe-templates

*aws ec2 describe-launch-templates*

**Create a new autoscaling group**

Specifications

create a new auto scaling group using the following settings:

* Name: awsgen-frontend-asg
* Launch template: awsgen-frontend-launch-template
* VPC: select the default VPC
* Availability zones & subnets: select the 2 AZs/subnets that you've used when you created the load balancer
* Attach to an existing load balancer (awsgen-frontend-tg | HTTP)
* Desired capacity: 2
* Minimum capacity: 1
* Maximum capacity: 5
* Scaling policies: None
* Tags:
  + Key: Name
  + Value: awsgen frontend

*aws autoscaling create-auto-scaling-group \*

*--auto-scaling-group-name awsgen-frontend-asg \*

*--launch-template "LaunchTemplateName=awsgen-frontend-launch-template" \*

*--vpc-zone-identifier "<subnet-1>,<subnet-2>" \*

*--target-group-arns "<target-group-arn>" \*

*--desired-capacity 2 \*

*--min-size 1 \*

*--max-size 5 \*

*--tags "Key=Name,Value=awsgen frontend"*

describe auto scaling groups

*aws autoscaling describe-auto-scaling-groups*