# Auto-Scaling

## VPC

## Subnet

## ELB

1. <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-subnets-commands-example.html>

aws ec2 create-vpc --cidr-block *10.0.0.0/16*



>aws ec2 describe-vpcs

Create subnets in this VPC:

>aws ec2 create-subnet --vpc-id vpc-0a91a30f0712e60ac --cidr-block *10.0.1.0/24*

>aws ec2 create-subnet --vpc-id vpc-0a91a30f0712e60ac --cidr-block *10.0.0.0/24*

2. Security group is required to set up INBOUND and OUTBOUND traffic:

>aws ec2 create-security-group --group-name *SSHAccess* --description "*Security group for SSH access*" --vpc-id vpc-0a91a30f0712e60ac

Output:

{

"GroupId": "sg-0b7f7fc6bd6b699f2"

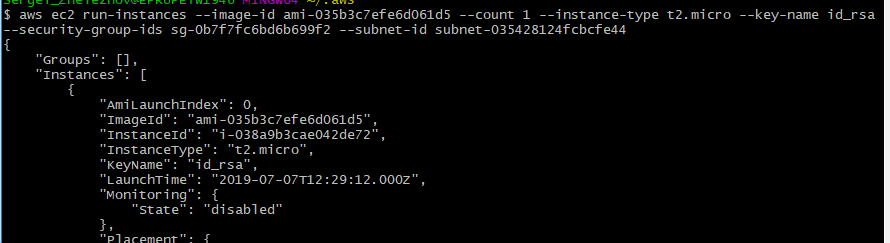
}

1. Allow SSH port to connect (open port)

aws ec2 authorize-security-group-ingress --group-id sg-0b7f7fc6bd6b699f2 --protocol *tcp* --port *22* --cidr *0.0.0.0/0*

1. Launch an instance into your public subnet, using the security group and key pair you've created.

>aws ec2 run-instances --image-id ami-035b3c7efe6d061d5 --count 1 --instance-type *t2.micro* --key-name *id\_rsa* --security-group-ids sg-0b7f7fc6bd6b699f2 --subnet-id subnet-035428124fcbcfe44



Note: see subnets

## >aws ec2 describe-subnets --filters "Name=vpc-id,Values=*XXXXXXXXXXXXXXXX*" --query 'Subnets[\*].{ID:SubnetId,CIDR:CidrBlock}' Elastic Load Balancer

* Create ELB

aws elb create-load-balancer --load-balancer-name crash-course-elb --availability-zones us-east-1a us-east-1b --listeners Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80

* Run instances

aws ec2 run-instances \

--image-id ami-97785bed \

--key-name aws-crash-course-us \

--instance-type t2.micro \

--user-data file://../compute-resources/scripts/instance\_bootstrap.sh

* Register instances

aws elb register-instances-with-load-balancer --load-balancer-name crash-course-elb --instances i-9961fe77 i-de024635

* Configure health checks

aws elb configure-health-check \

--load-balancer-name crash-course-elb \

--health-check Target=HTTP:80/index.html,Interval=10,Timeout=5,UnhealthyThreshold=2,HealthyThreshold=2

* Adjust settings

## Auto Scaling Group

* Create Launch Configuration
  1. Using custom AMI
  2. Using user data script

aws autoscaling create-launch-configuration \

--launch-configuration-name crash-course-lc \

--image-id ami-97785bed \

--key-name aws-crash-course-us \

--instance-type t2.micro \

--user-data file://../compute-resources/scripts/instance\_bootstrap.sh

* Create auto scaling groups

aws autoscaling create-auto-scaling-group \

--auto-scaling-group-name crash-course-ag \

--launch-configuration-name crash-course-lc \

--min-size 1 --max-size 3 \

--load-balancer-names crash-course-elb \

--availability-zones us-east-1a us-east-1b

* Add scale policies
* Test

ab -n 100000 -c 10 "http://crash-course-elb-687564412.us-east-1.elb.amazonaws.com/"

aws autoscaling delete-auto-scaling-group --auto-scaling-group-name crash-course-ag

aws autoscaling delete-launch-configuration --launch-configuration-name crash-course-lc

aws elb delete-load-balancer --load-balancer-name crash-course-elb

or use Postman to run 1000 requests in tests