E91 Cloud DevOps: Fall 2018

Assignment 1

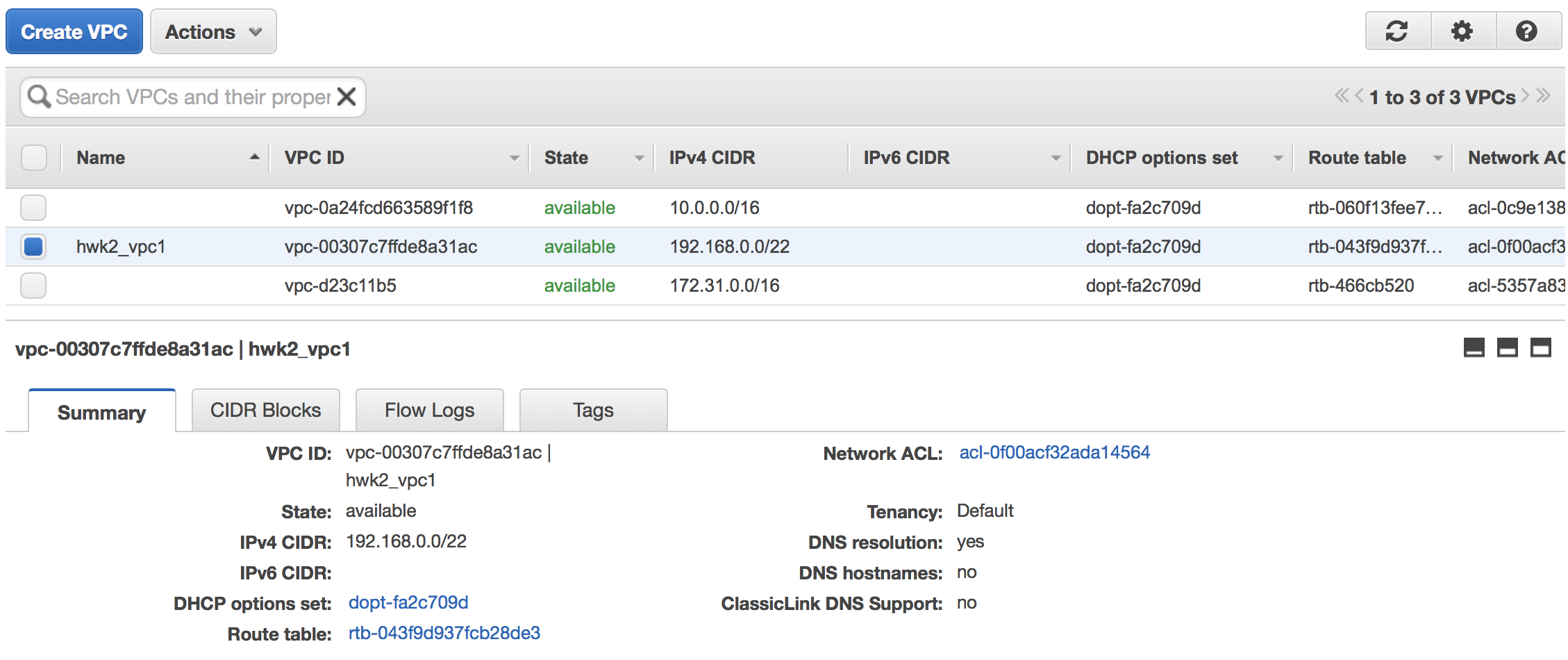
Stephen Akaeze

<https://code.harvard.edu/sta283/cscie-91_sta283>

Assignment Instruction Manual: <https://canvas.harvard.edu/courses/53026/files/6620943/download?wrap=1>

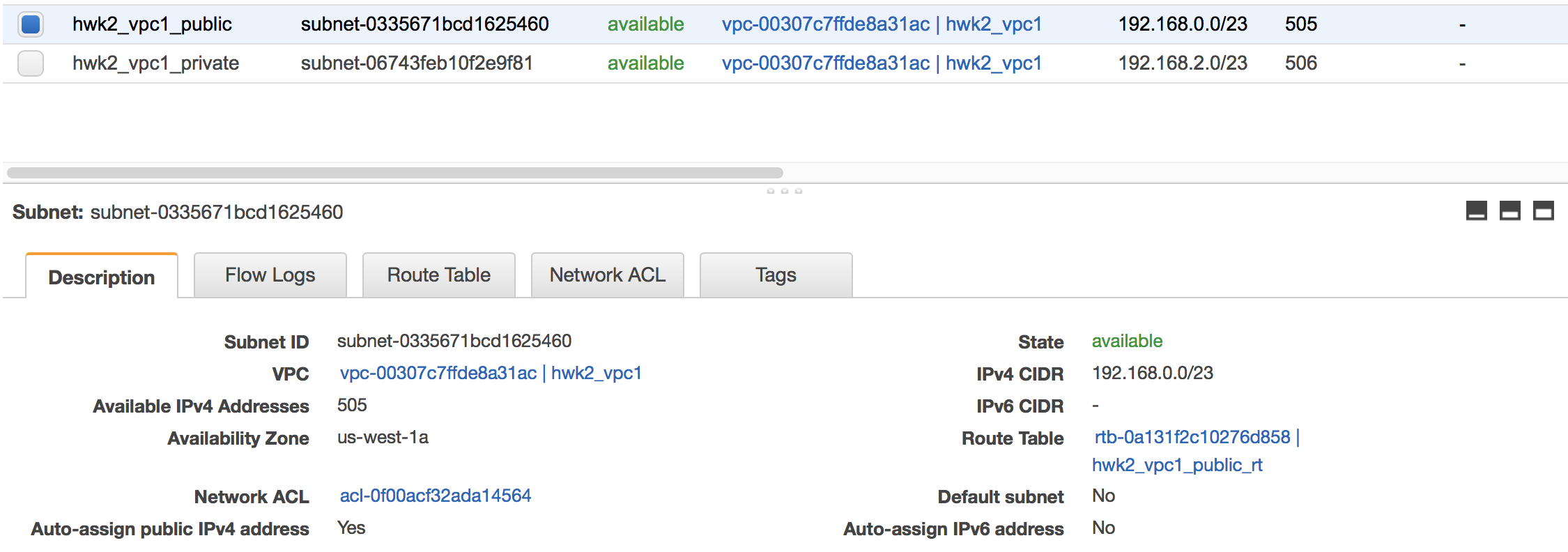
**Problem 1:**

1. Setup a VPC with appropriate tag/name of CDR of 192.168.0.0/22
   1. **Answer:** There are 1024 possible IPs in this CIDR of the selected VPC below
   2. **Result: The resulting VPC details are captured below**

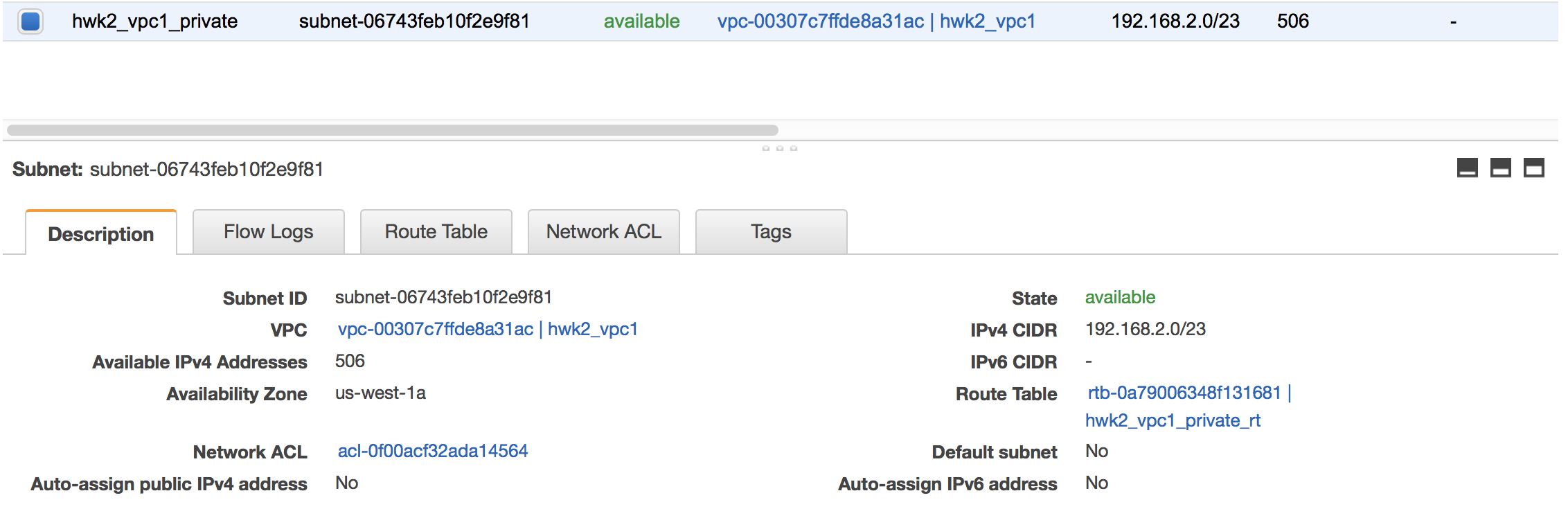


1. Split the VPC CIDR equally amongst two subnets tagged, Public and private
   1. **Result: The resulting public and private subnet details are captured below**

Public Subnet



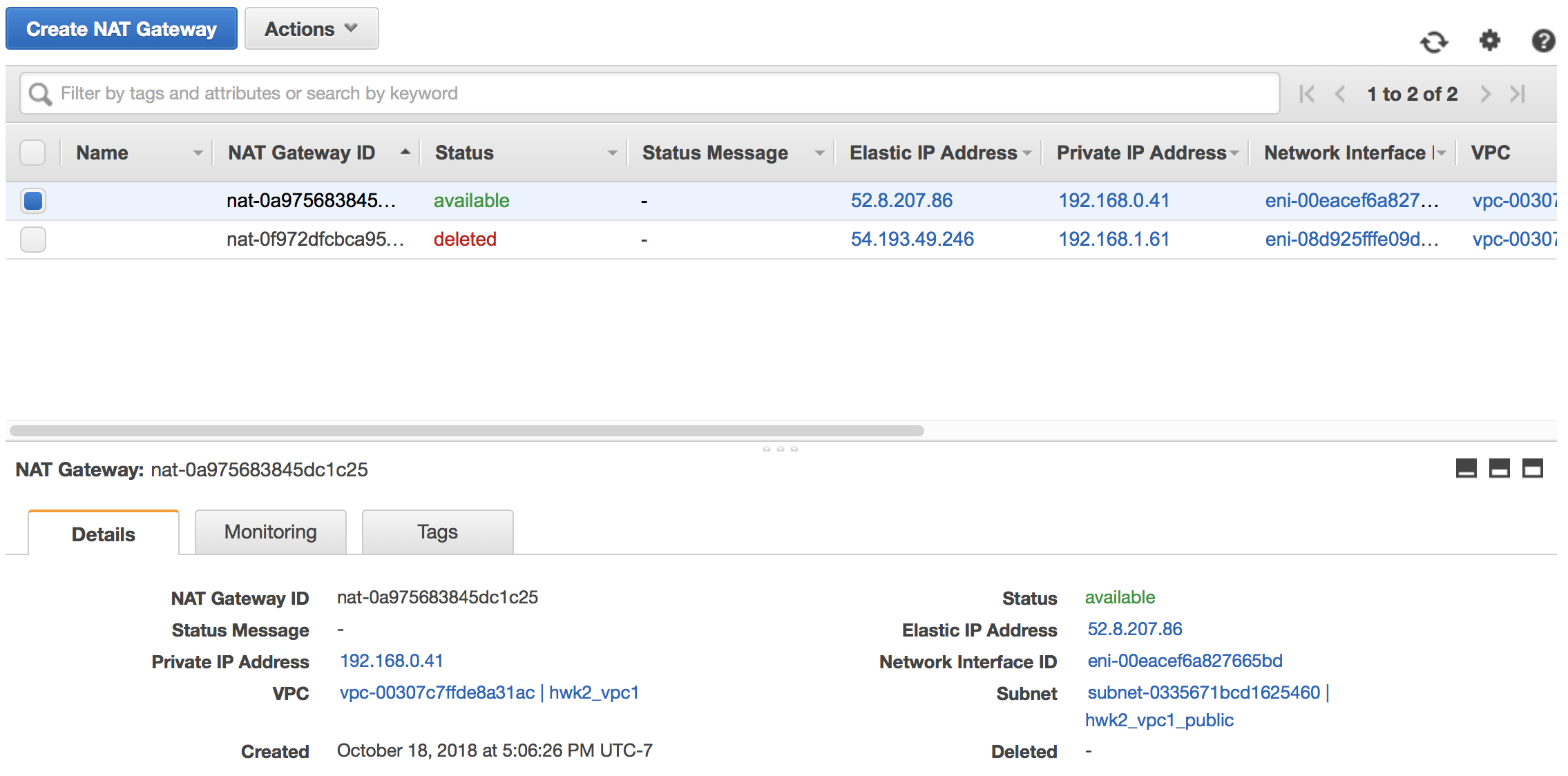
Private Subnet



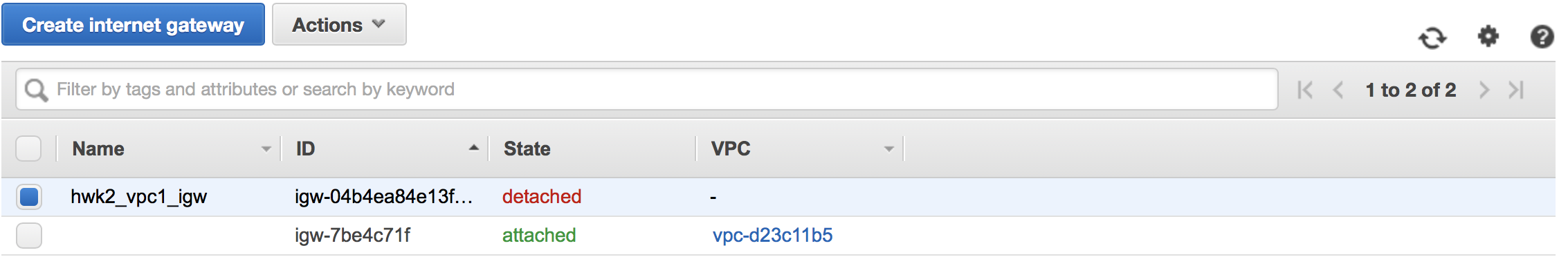
1. Create a Nat gateway and Internet gateway

**Result: The resulting NAT and Internet gateway details are captured below**

Nat Gateway



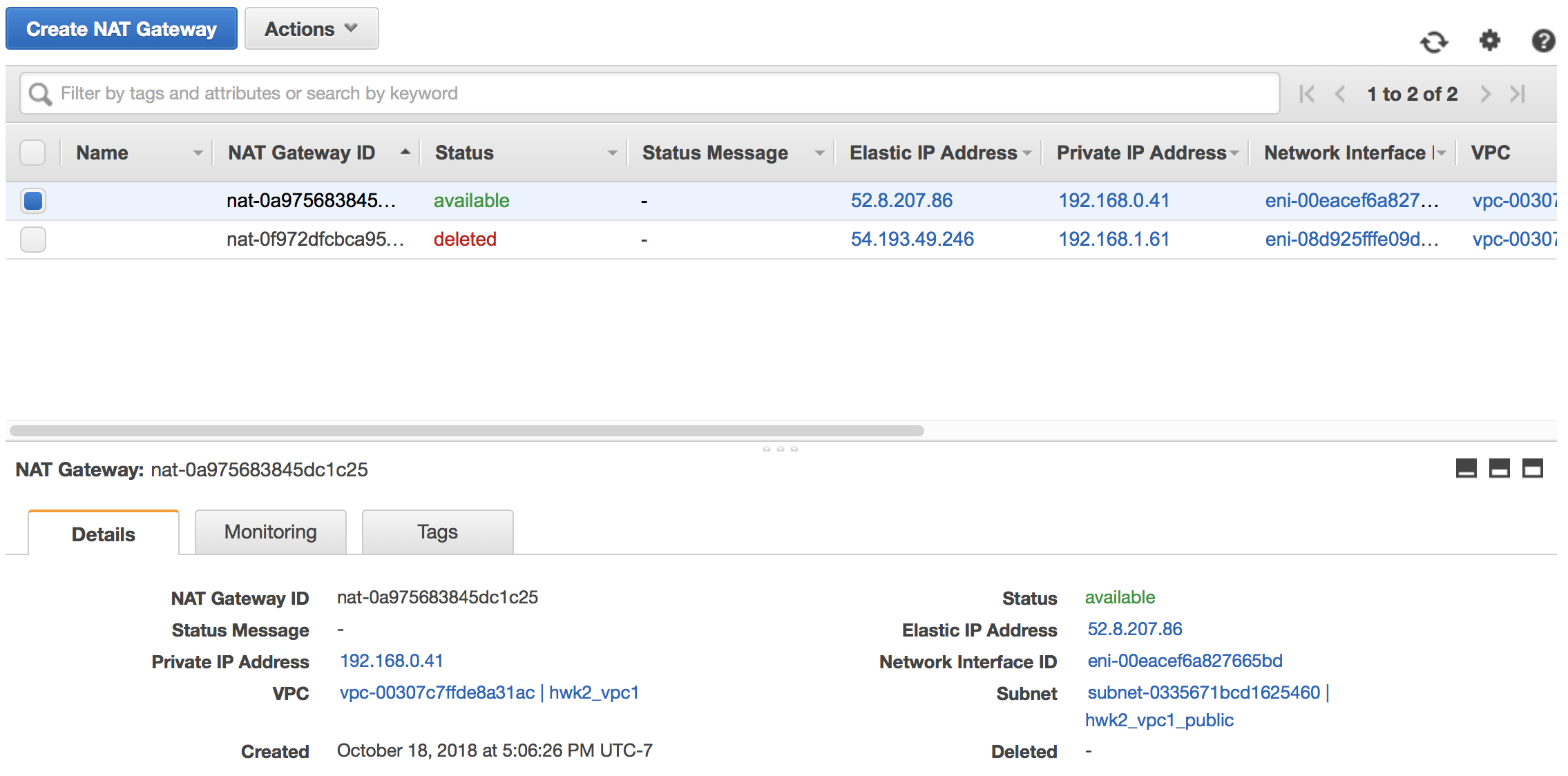
Internet Gateway



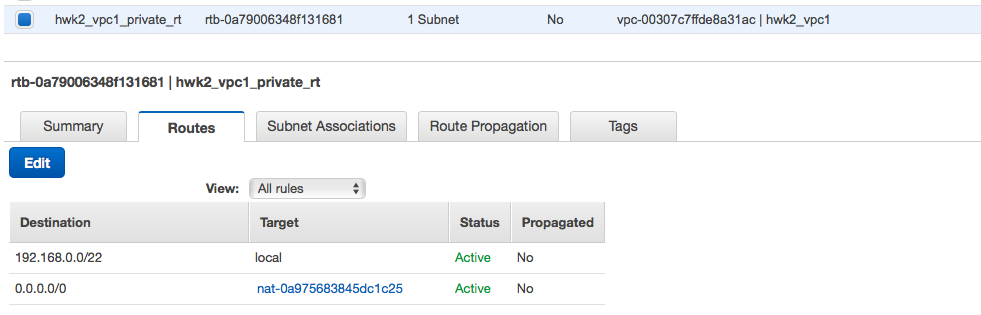
1. Setup routes in route table so private subnet can access internet via NAT while public subnet has access via IGW

**Result: The NAT gateway, internet gateway, routing tables and subnet configuration details are shown below**

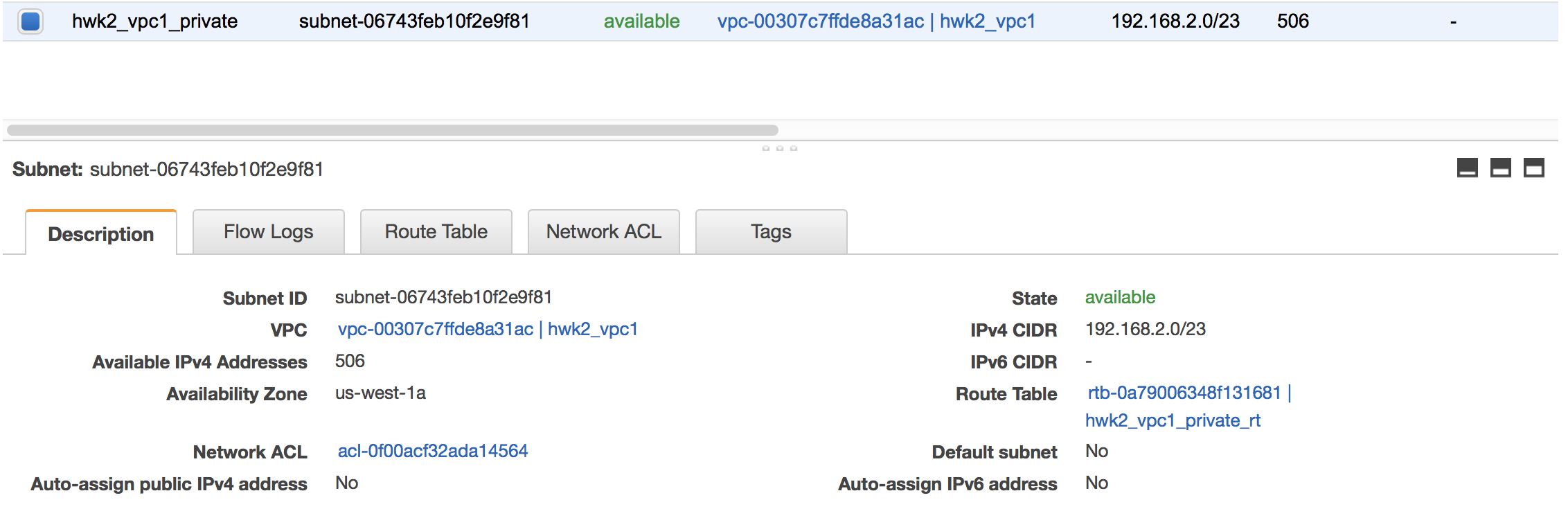
Nat gateway config showing 192.168.0.41 IP in public subnet



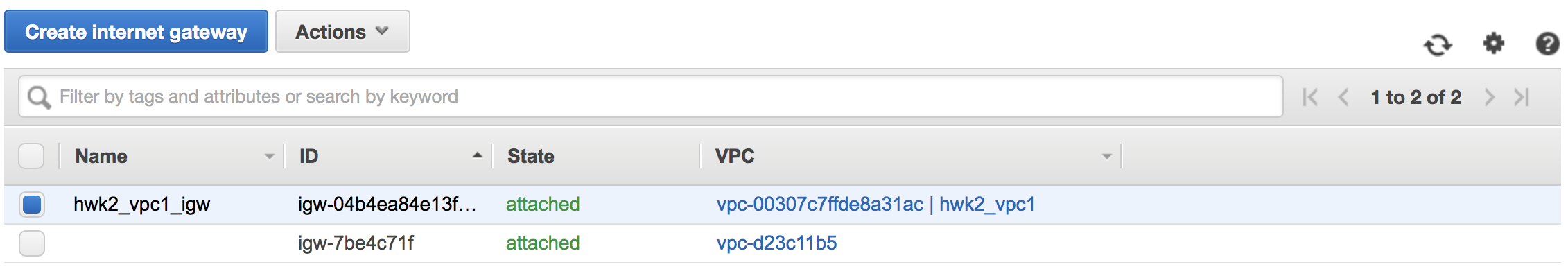
Private routing table showing internet request redirection via Nat gateway



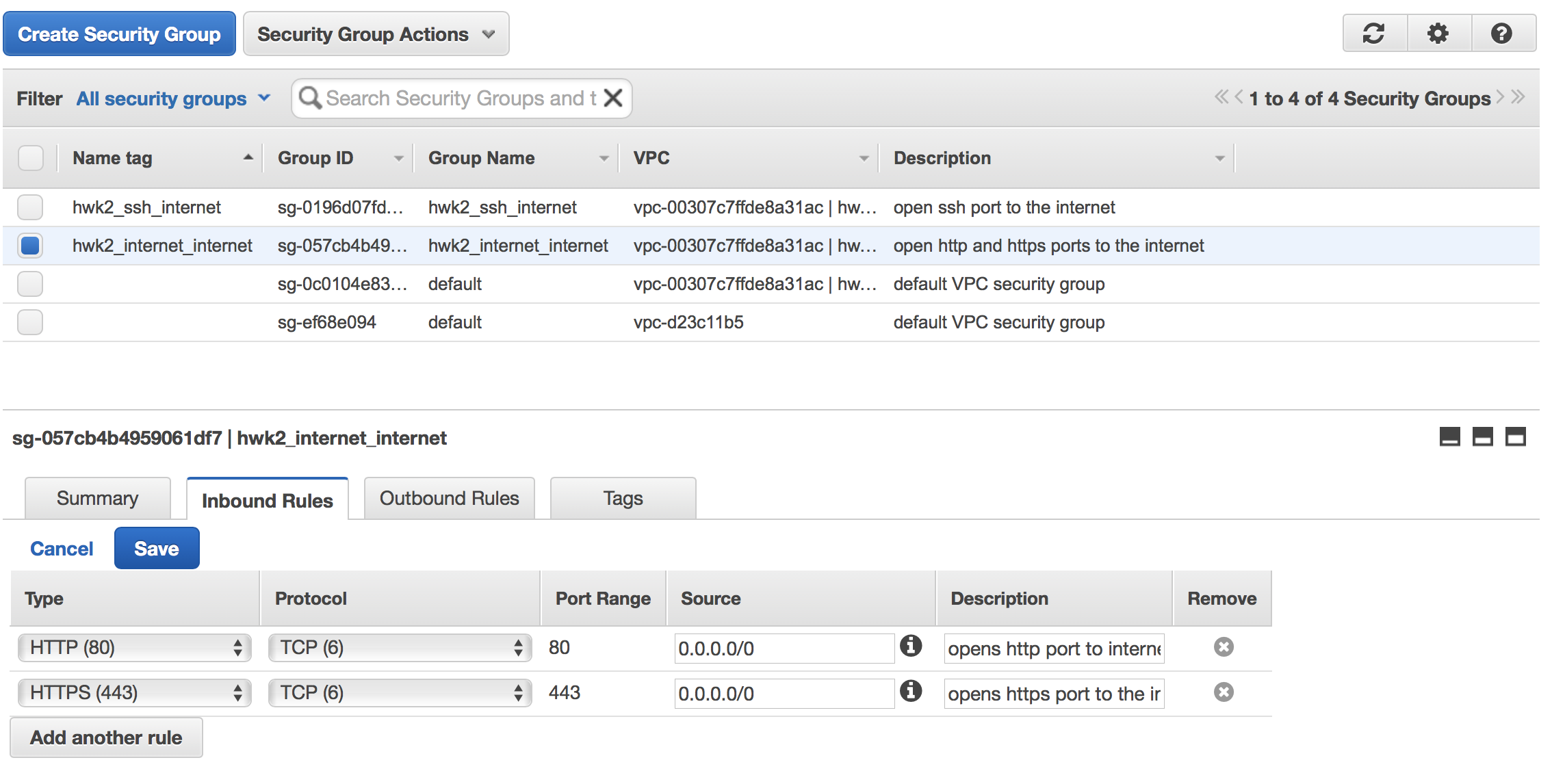
Private subnet showing attached private routing table



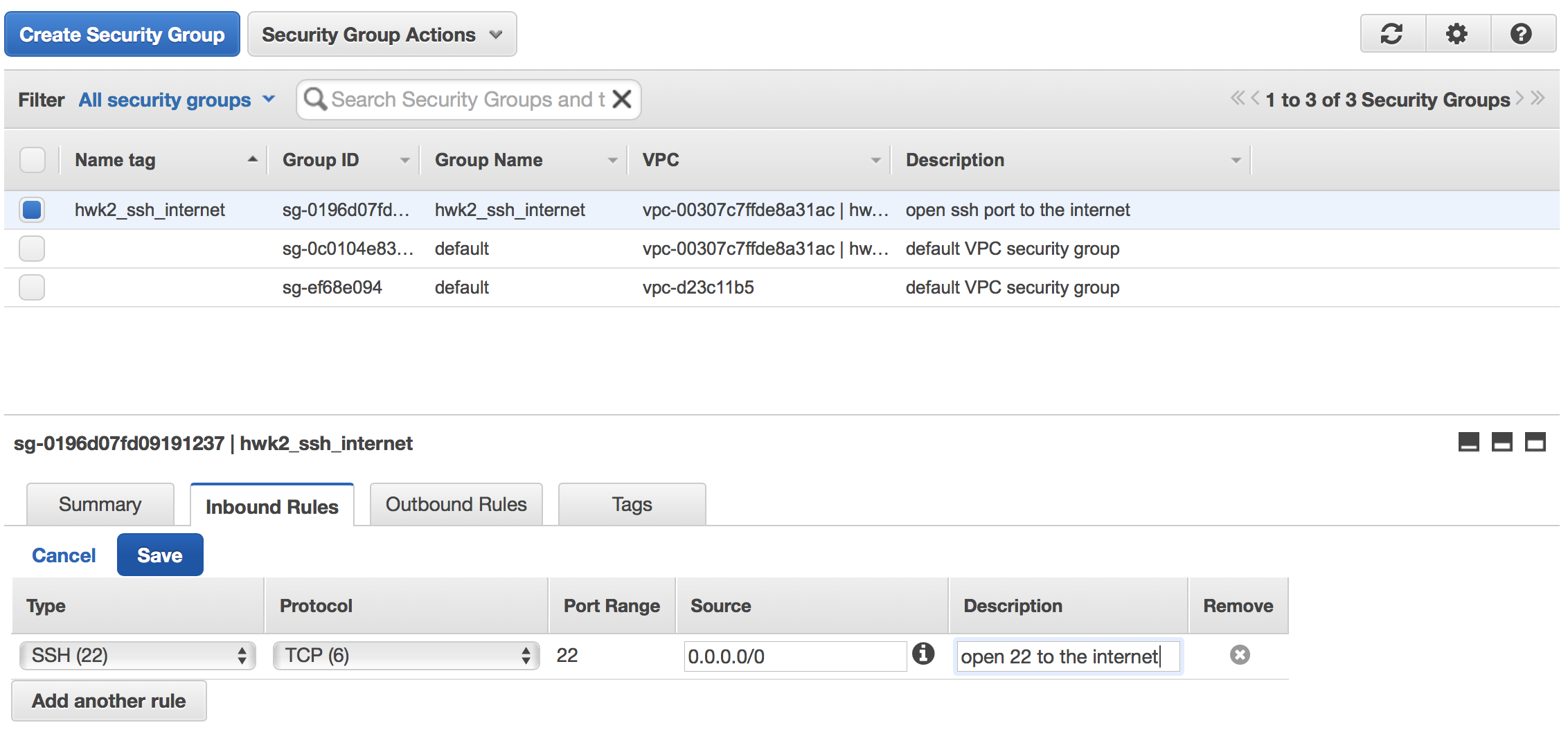
Internet gateway showing attachment to the VPC



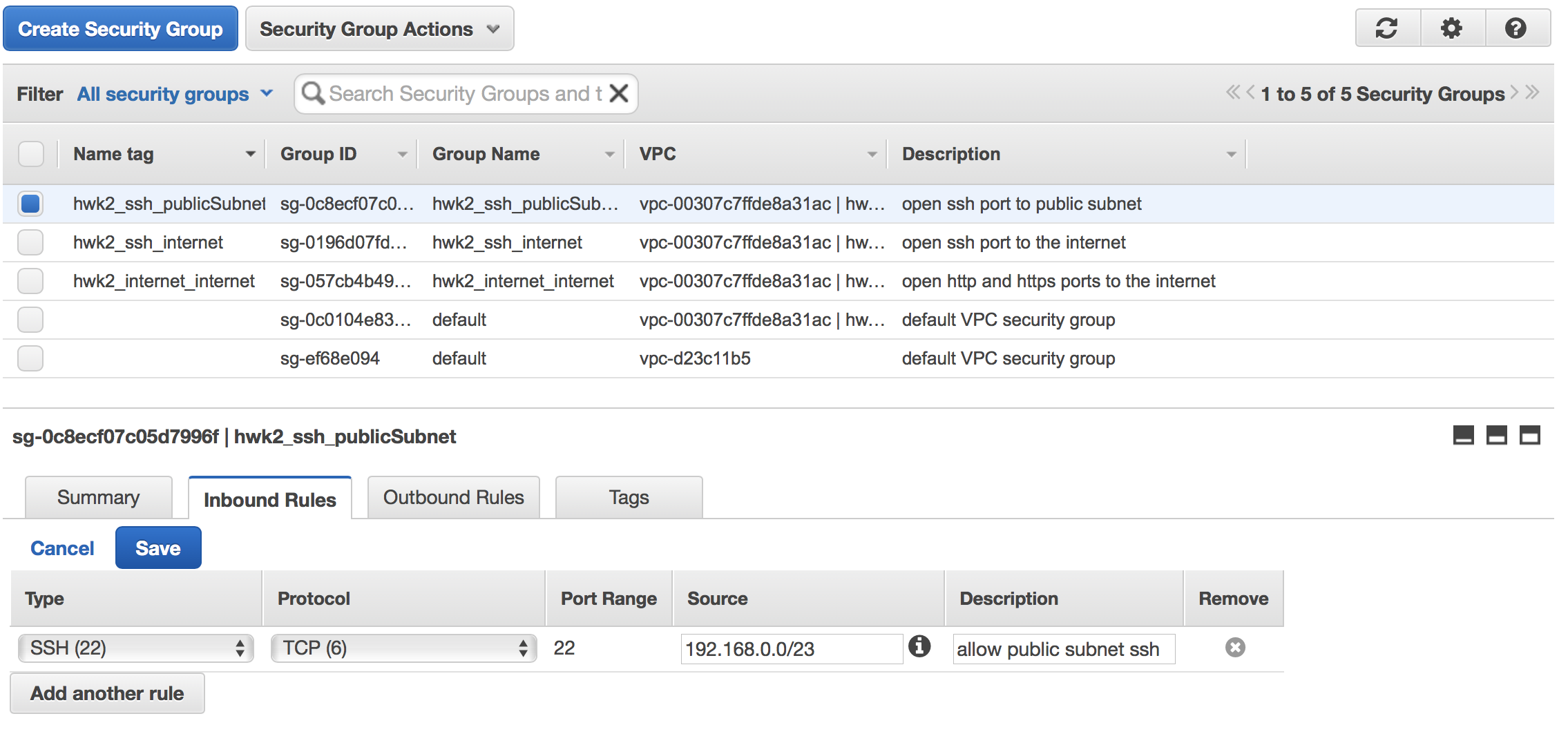
Security group for public instance for allowing inbound http and https connections



Security group for public instance for allowing inbound ssh connections



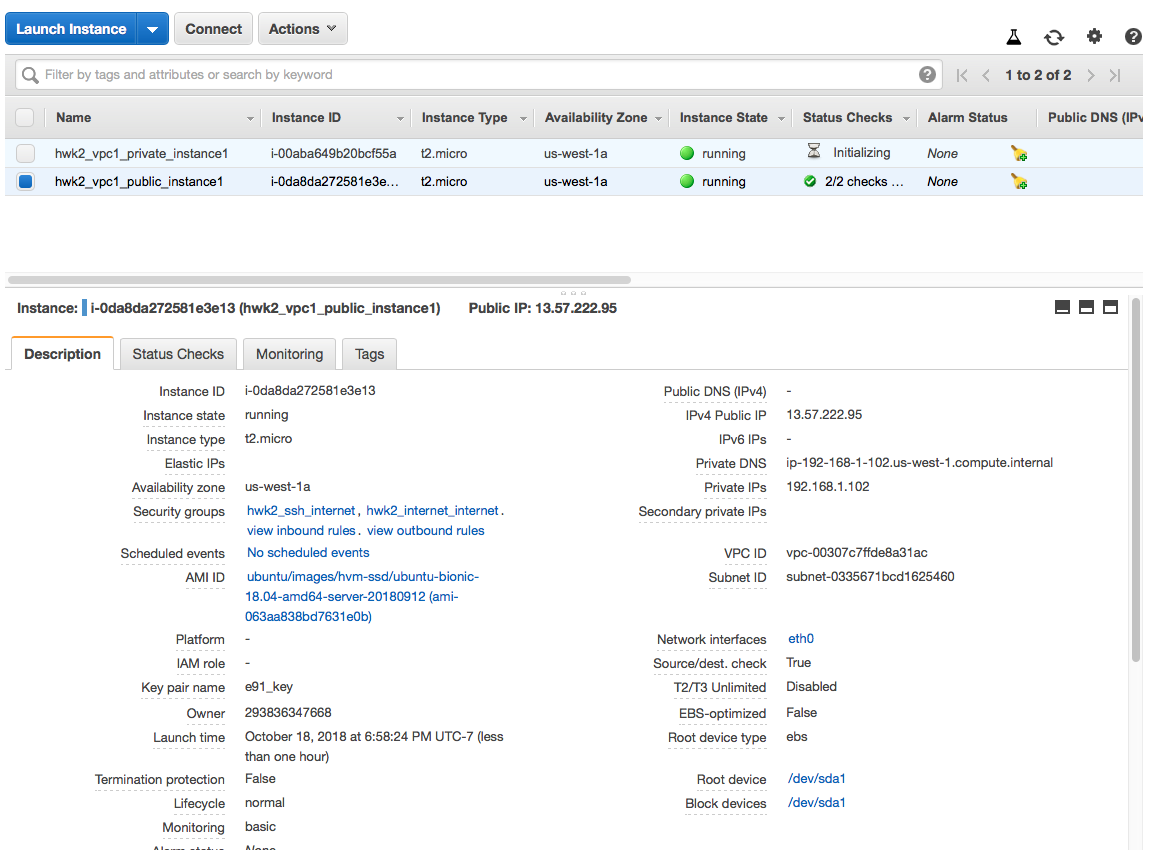
Security group for private instance for allowing inbound ssh connections from public subnet



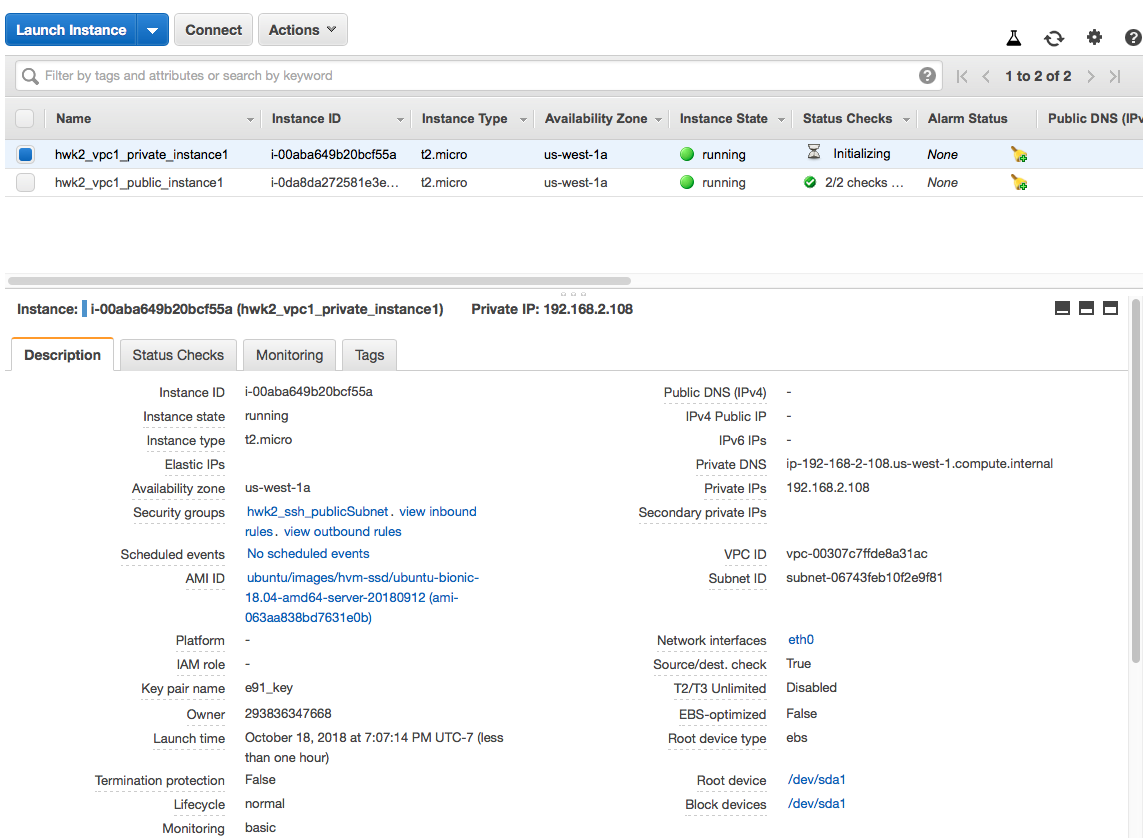
1. Create an EC2 instance in both public and private subnet. Then demonstrate locking when connecting to Internet from both instances

**Result:**

Public Instance showing security groups, public IP, private IP etc.



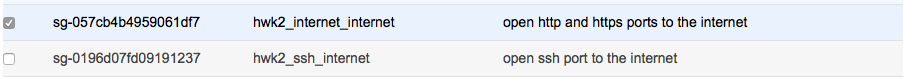
Private Instance showing security group, private IP etc.



**Locking demonstration:**

**Public Instance**

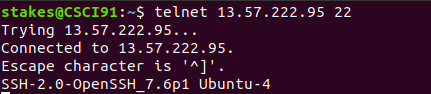
when ssh port security group is disabled on the public instance via the AWS console



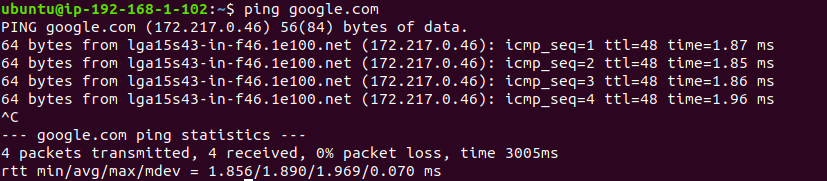
There was no telnet response from local VM to public instance



However, when the ssh port security group on the public instance is reenabled, the telnet command indicates ssh connectivity via the public instance IP



After accessing the public instance, the public instance also demonstrates internet connectivity via the “ping” command to google.com

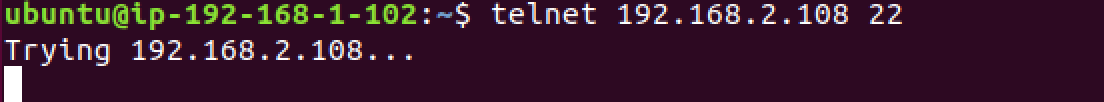


**Private Instance**

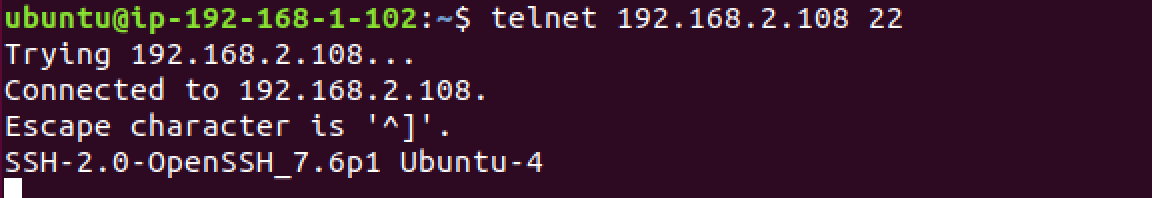
When the ssh port to public subnet security group is disabled on the private instance via AWS console



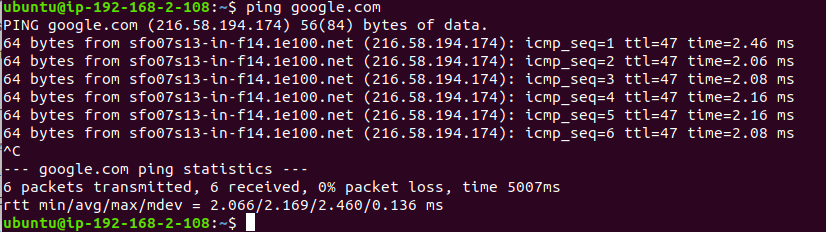
There is no telnet response from the private instance



However, when the ssh port to public subnet is reenabled, the telnet command indicates ssh connectivity via the private IP of the private instance.

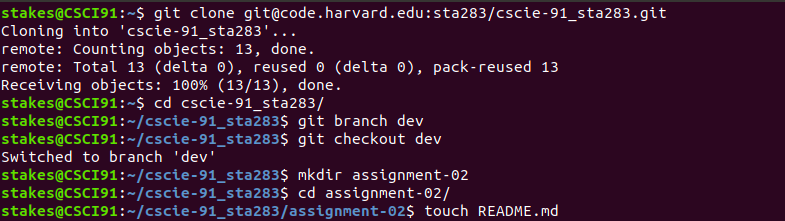


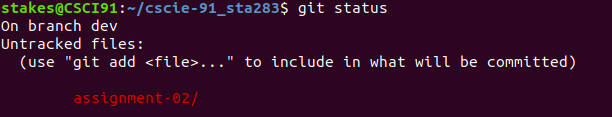
The private instance also demonstrated internet connectivity via the pin g command

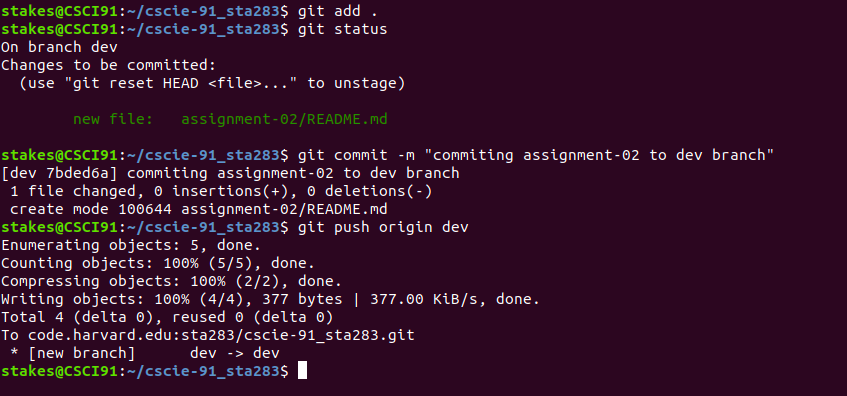


**Problem 2:**

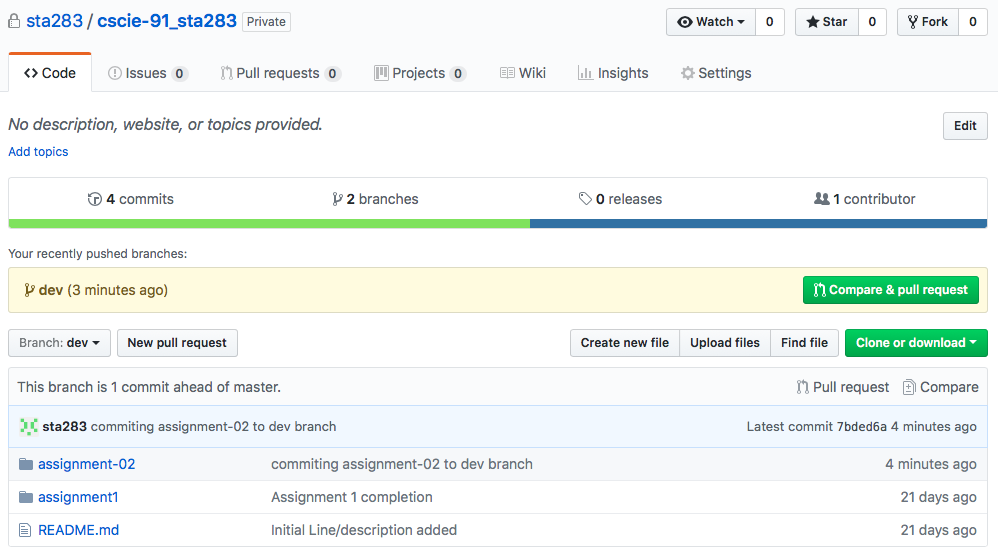
All commands were executed as captured in the snapshots below





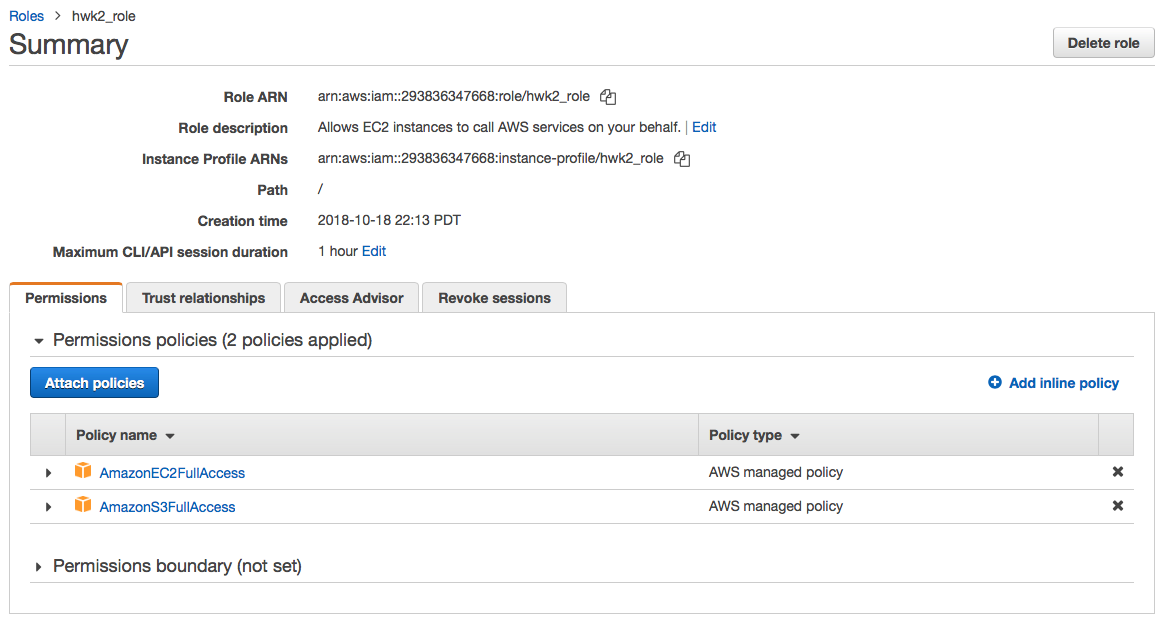


The following are the GitHub snapshots demonstrating the successful “dev” branching, commit and push along with the new directory and file. Please visit <https://code.harvard.edu/sta283/cscie-91_sta283> for more information

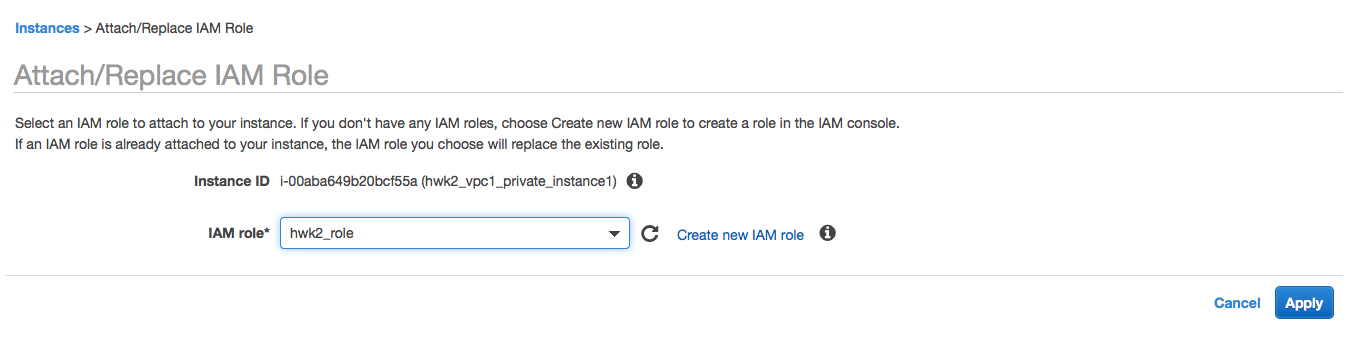


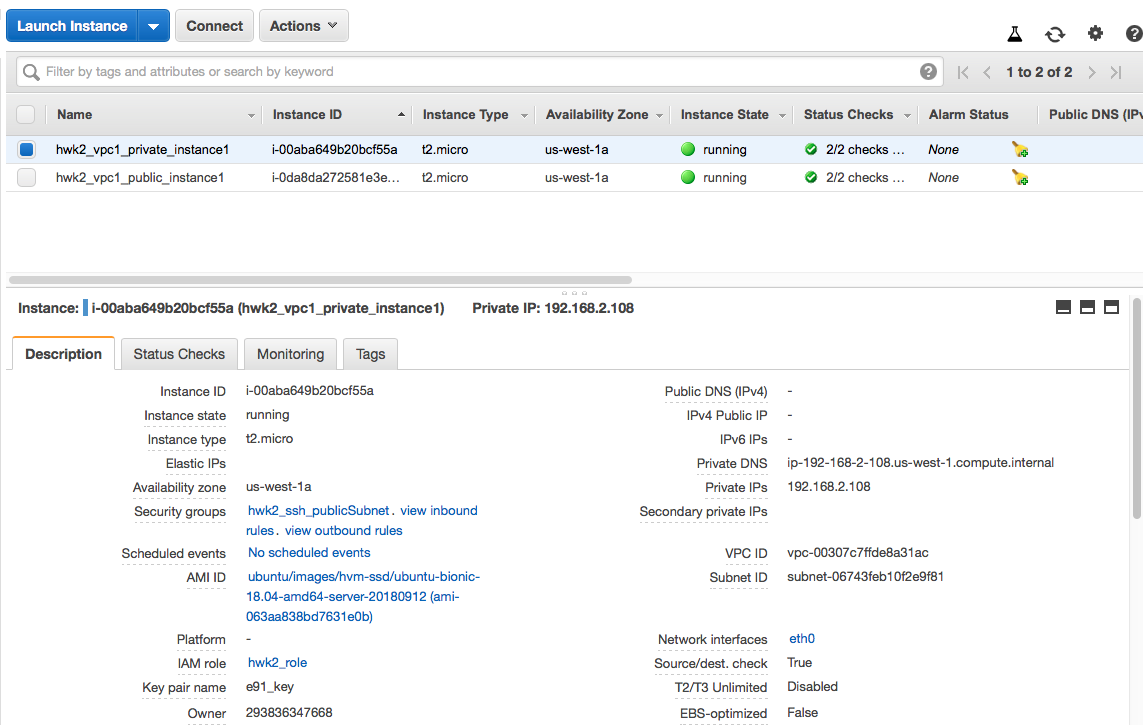
**Problem 3:**

1. Create an IAM Role with permission to fully manage EC2s and S3
   1. **Result:**



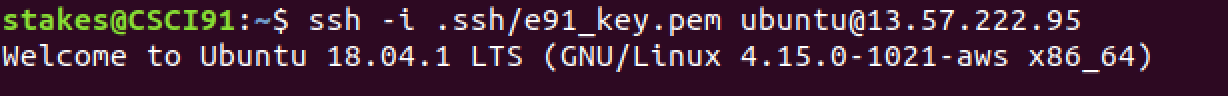
1. Assign role to the private instance
   1. **Result: Below shows the IAM role creation and attachment to the private instance.**





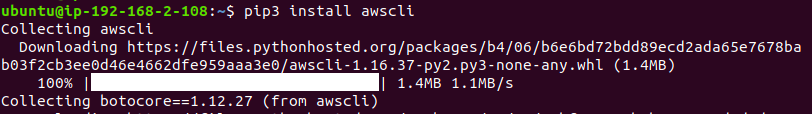
1. Log into the private instance.
   1. **Result: As shown below, the private key is copied to the public instance. Using ssh, I accessed the public instance. From the public instance and using ssh, I accessed the private instance**



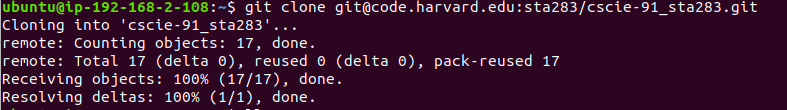


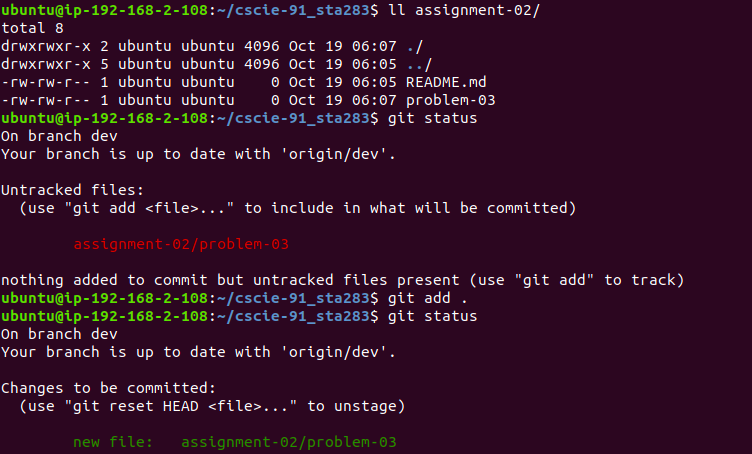


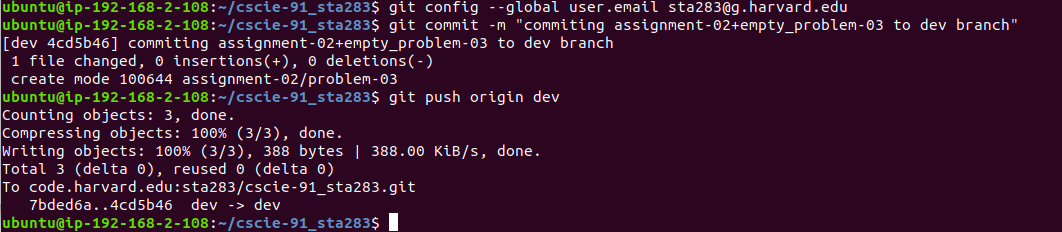
1. Install AWSCLI
   1. **Result: Used “pip3 install awscli” to install awscli. Then used “aws configure” to set my availability zone to us-west-1a**



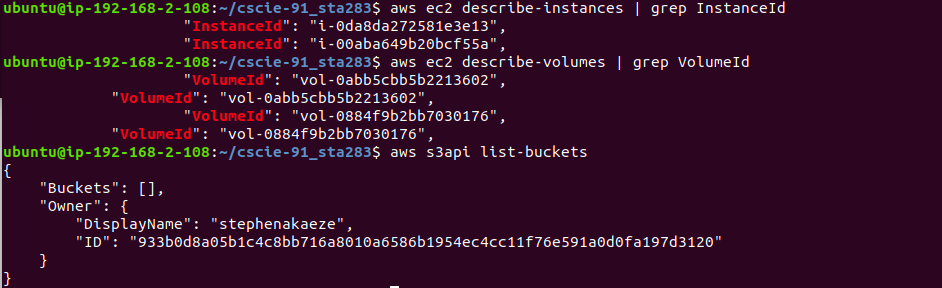
1. Clone your repository and switch to the branch dev, and create a file inside assignment-02, name it problem-03, commit and push the file to the dev remote.
   1. **Result: The instructions were all executed as shown by the screenshots below**



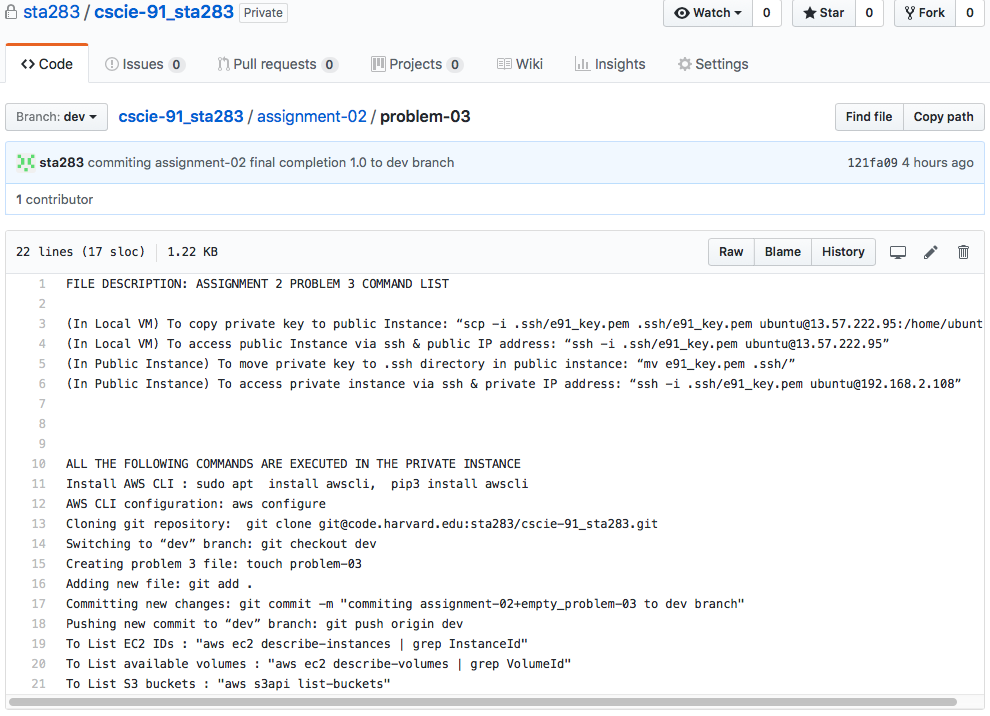




1. Use AWS CLI to list the IDs of EC2s, available volumes and S3 buckets
   1. **Results: the executed commands and results are shown below**



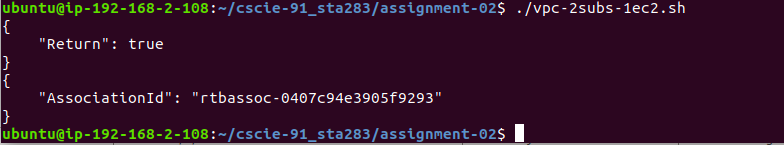
1. Capture all above commands in problem 3 file
   1. **Results: the commands are captured in the file as shown below**



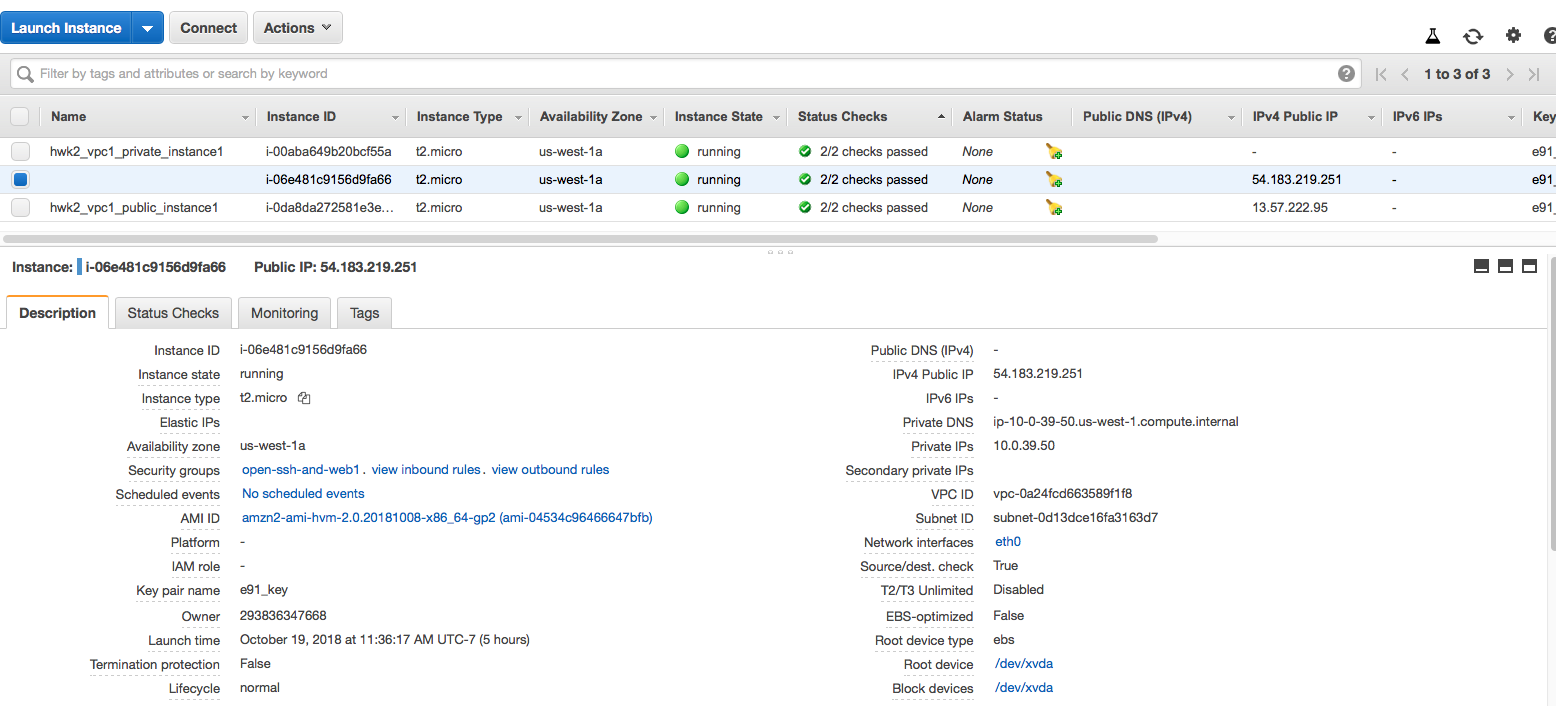
**Problem 4:**

Results: The original code from class came with bugs which were fixed and used the generate the required and specified network. Then it was stored as vpc-2subs-1ec2.sh in assignment-02 and pushed to dev. Below are individual proofs that the script was successful. Please visit <https://code.harvard.edu/sta283/cscie-91_sta283/blob/dev/assignment-02/vpc-2subs-1ec2.sh> to view the script details

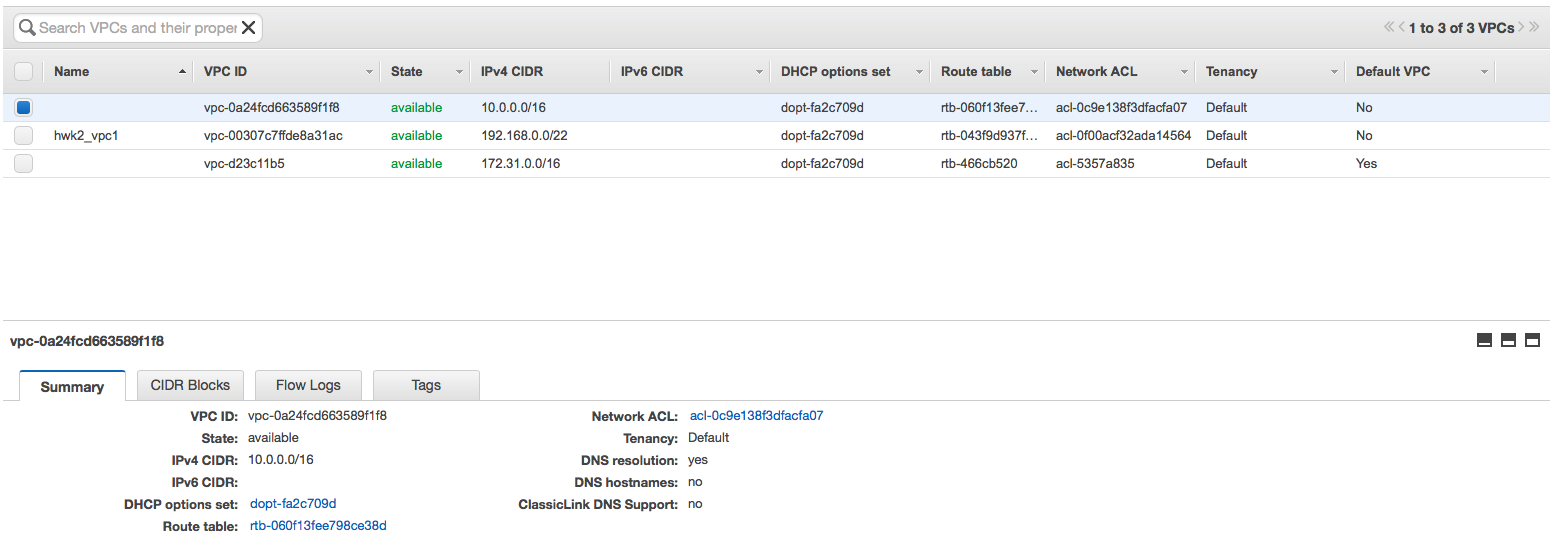
Console output after running script



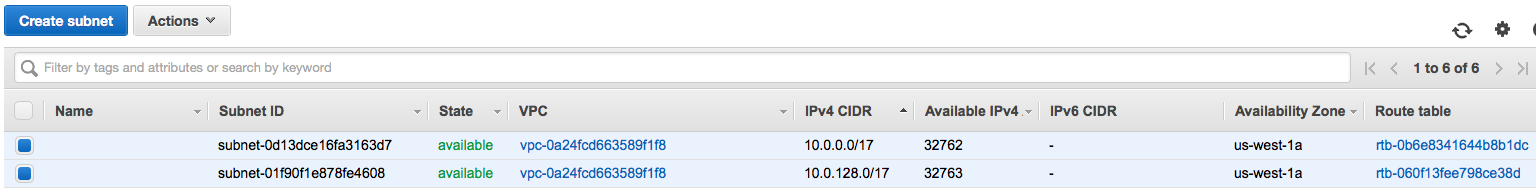
Instance created from script



VPC created from script



Subnets created from script



vpc-2subs-1ec2.sh @ code.harvard.edu

