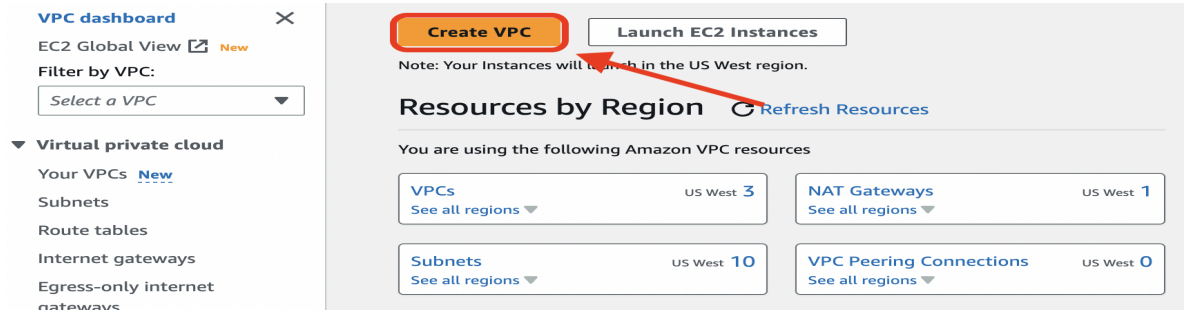
Straight Forward AWS EC2 Web App Secured using Bastion Host

Hello! Today I will be showing you how to securely deploy a web based application on Amazon Web Services's (AWS) Elastic Compute Cloud (EC2). To follow along it is required that you have credits on an AWS account. We create the following in order to deploy our app:

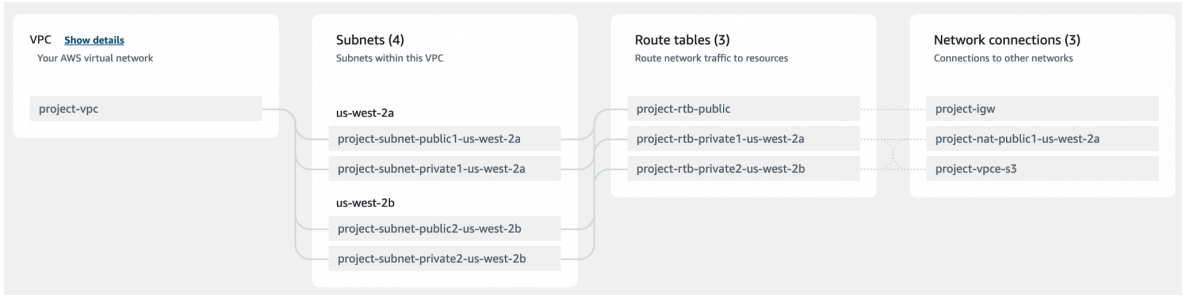
- VPC (virtual private cloud)
- Security Groups
- EC2 instances
- Domain & SSL certificates
- Load Balancer

Virtual Private Cloud

We need a VPC in order to ensure the security of our deployment. Essentially, the VPC allows us to work inside a private network so that our internal communication is completely secure.
To start, navigate to the AWS console home page and select the region you would like to deploy to. Now click on VPC and it should take you to the VPC dashboard. Click "Create VPC" as shown below:



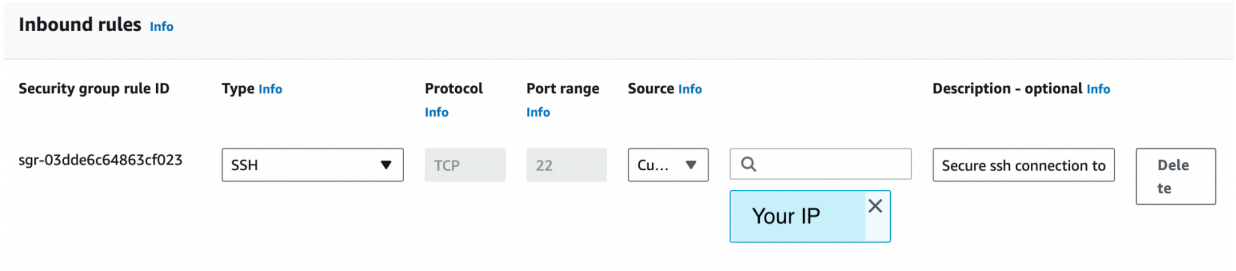
Under "resources to create" select "VPC and more". You should now see the VPC preview on the right of the options. Name your VPC and select auto-generate so things don't get mixed up. Leave the IP section as default values and navigate to the availability zone (AZ) where we want to select at least two, in this tutorial I will be using two. Additionally, select two public and two private subnets. Finally add one NAT gateway. The VPC preview should now look like this:



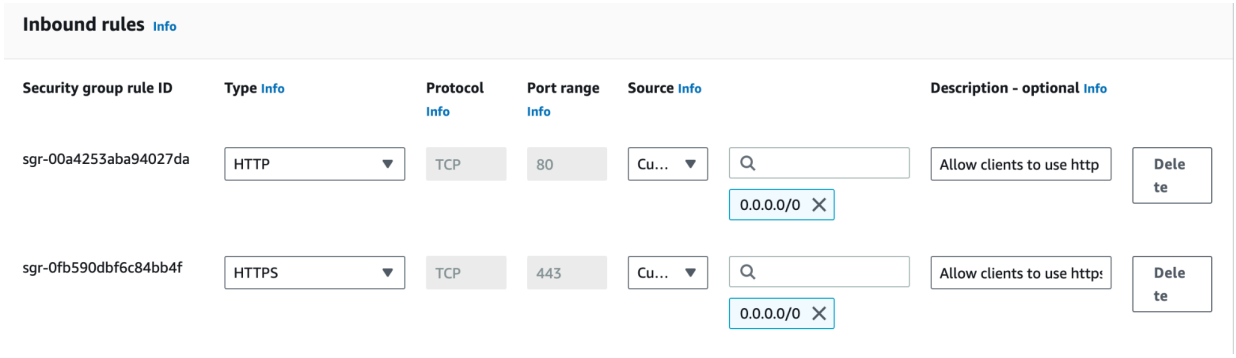
Put none for VPC endpoints and leave the DNS options enabled. Double checking that the previews line up, scroll to the bottom and click "Create VPC". We have now completed the first step by creating our virtual private cloud.

Security Groups

Security groups are the rules that dictate what protocols and from where those protocols are allowed through specific parts of our deployment. These allow us to do things like fully secure our bastion host and allow communication between the load balancer and private instances. Now we will set up the security groups for each of the components needed for deployment. The first security group we will create is the one for our bastion hosts. Each public subnet across the AZs will have an EC2 instance that we will use to connect to the EC2 instance on the private subnet of that AZ. Thus, the public instance is a bastion host because it is our fortress that we must harden and keep secure; since once access is gained to the bastion access is granted to everything else. Therefore, the security group for our bastion hosts must be very stringent. We need to allow the ability to SSH into the bastion so we can access it and the private subnet behind it but if anyone else gets this ssh ability it would be a complete security breach. So, the group must allow SSH but only from trusted IP addresses; take a moment to find your IP (assuming you're in a secure network). Then navigate to "Security groups" under "Security" in the VPC dashboard and select "Create security group". Give it a name along the lines of bastion or public and add the following inbound rule, leave the outbound rules to allow anything, and click "Create security group":

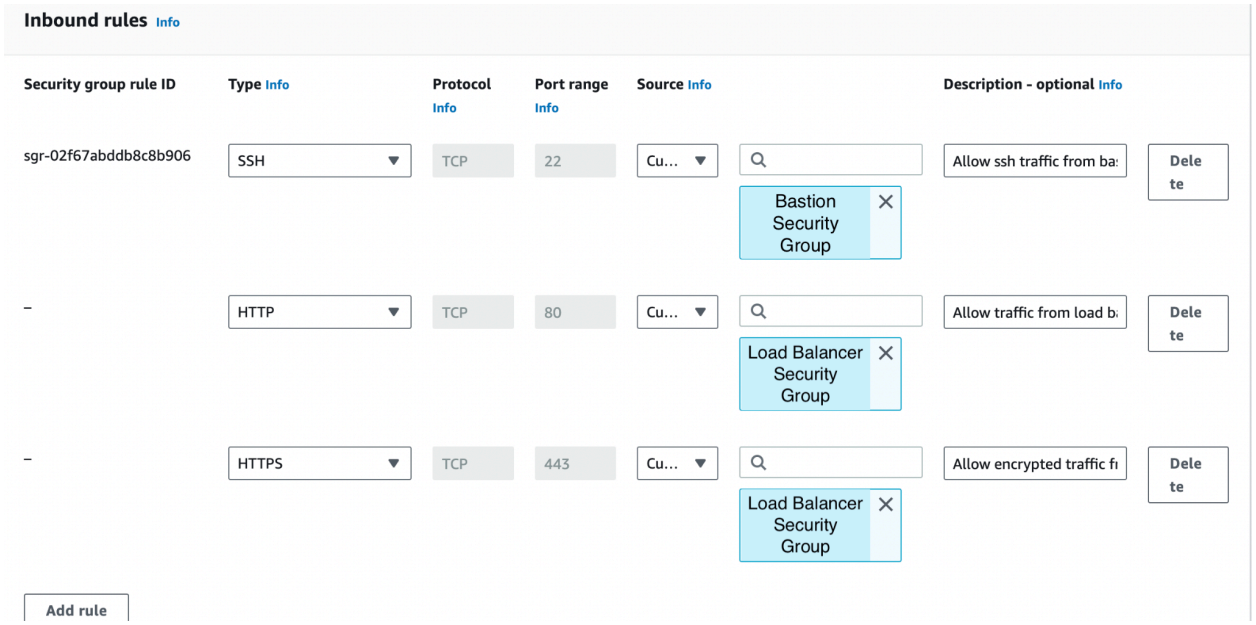


Next we will create the security group for the Load Balancer. The Load Balancer is what the clients of our web application will be connecting to so we need to accept both HTTP and HTTPS from anywhere, again allowing any outbound traffic, and create the group as bellow:



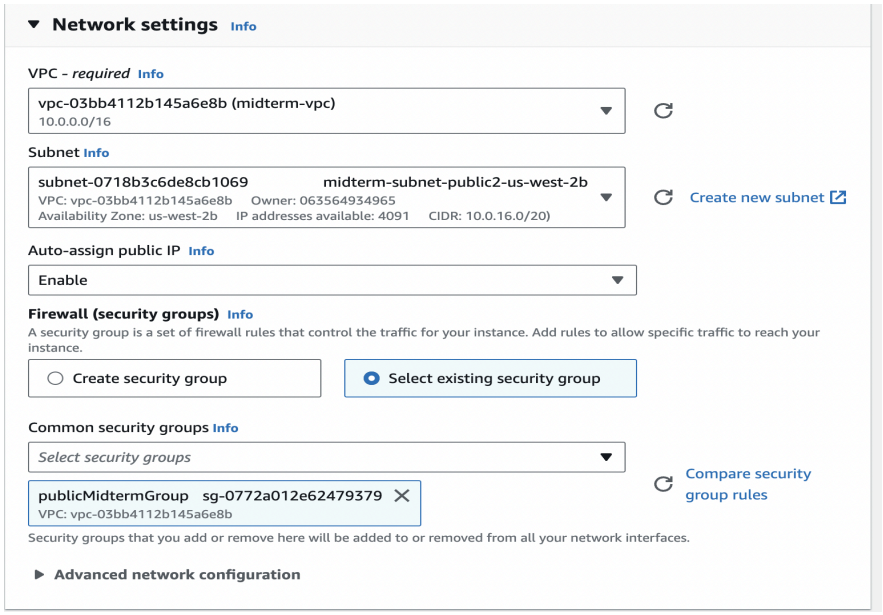
Finally we will create a security group for our private instances. The private instances need to accept SSH from the bastions and HTTP/HTTPS from the load balancer. This time we will not

be using IP addresses for the sources but instead now we can use the security groups we just created. Once again we allow all outbound traffic, and our inbound rules should look as follows:

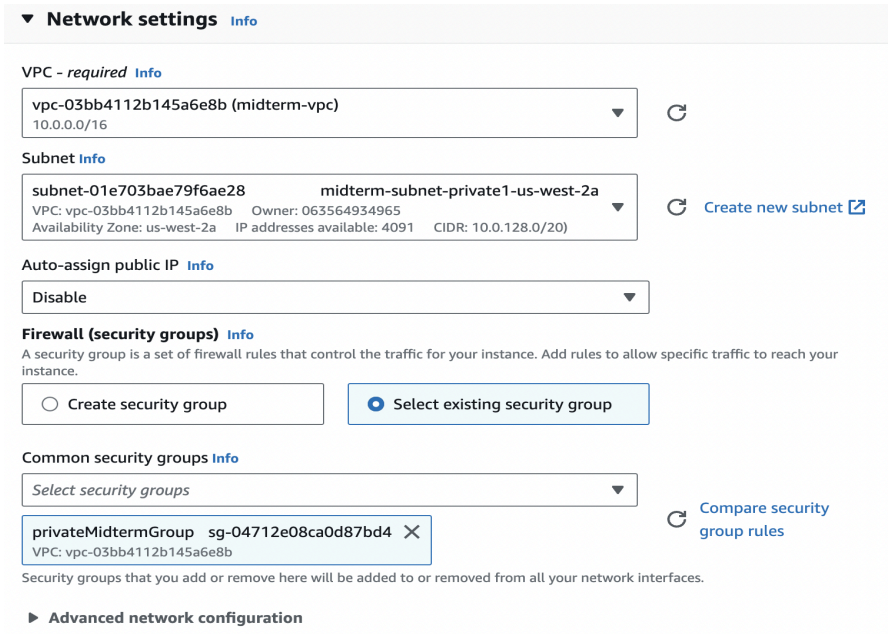


Instances

Instances are the actual virtual machines (VMs) that we can SSH into and deploy code onto. Now we will actually launch the instances we talked about in the security group section. We will be launching four instances total, two public and two private. Let's start with the public. Navigate to the EC2 Dashboard and select "launch instance". Give your instance a name (make sure it is clear it is the bastion/public) and select an operating system (OS). This tutorial will be using Ubuntu. Select an instance type based on your needs and create a key pair. In the network settings first click edit in the top right. Then we want to select the VPC we created under "network". Then select one of the public subnets for the subnet section. Enable auto-assign public IP so our bastion has a public IP. Finally for the firewall section choose "select existing security group" and select the bastion/public security group you created. It should look like this:



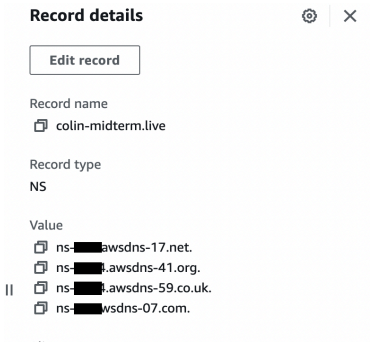
Configure the amount of storage needed for your application and then click “launch instance”. Repeat this process again but change the subnet to the other public subnet. Now we have both our bastion hosts set up. Now to launch the private instances repeat the same process once again up until the network settings where we now must choose a private subnet and disable “auto-assign public IP”. It should look like this:



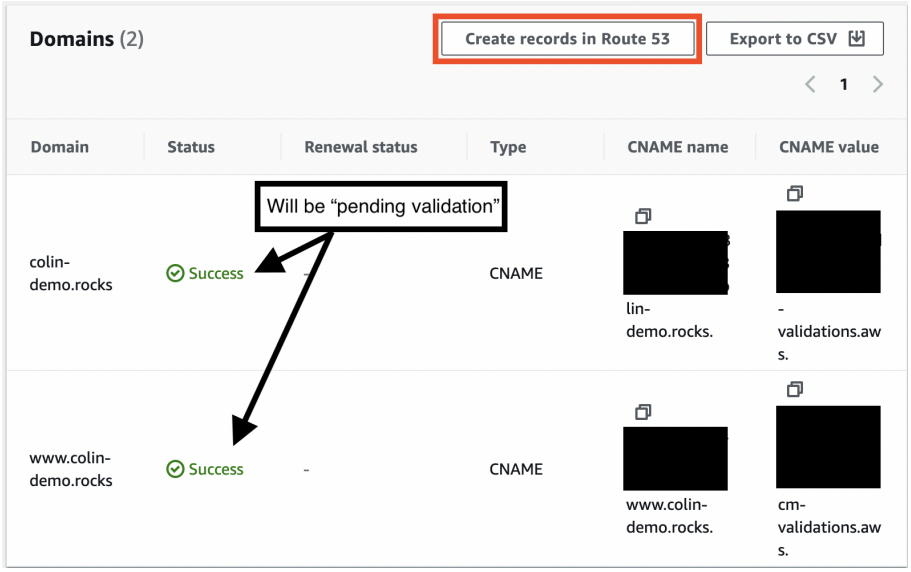
Repeat this process once again for the other private subnet. After launching all four we now have all of the instances we need up and running with the correct security.

Domain & SSL

In order to deploy our web application we will need a domain so people can find it. Use any domain creation service to create a domain with your desired name. Once you have a domain navigate to the Route 53 dashboard in AWS. Select “hosted zones” on the left hand side and then click “create hosted zone”. Paste the domain you just created into the “domain name” section, give it a description, and make sure to select “public hosted zone” under “type”. Click “create hosted zone”. Now navigate to the hosted zone you just created and copy all four nameservers (NS) in the NS record. The NS record should look something like this:



Return to the service you created your domain with and navigate to the section that allows you to manage nameservers. Remove all of the nameservers that the domain came with and replace them with the nameservers acquired in the NS record; make sure to save the changes. Now back on AWS navigate to the AWS Certificate Manager (ACM). Select “request certificate” and then select “request a public certificate” under “certificate type”. Under “domain names” first paste your domain name; then click “add another name to this certificate” and put “www.” followed by your domain name again. For “validation method” we want “DNS validation” and leave the “key algorithm” as RSA. Now select “request”, we should now have a new certificate with the status of pending validation. Click on the new certificate to see its details. Scroll down to the “domains” section where you should see both your domain and www.yourdomain. Now click the “create records in Route 53” button in the top right as shown below:

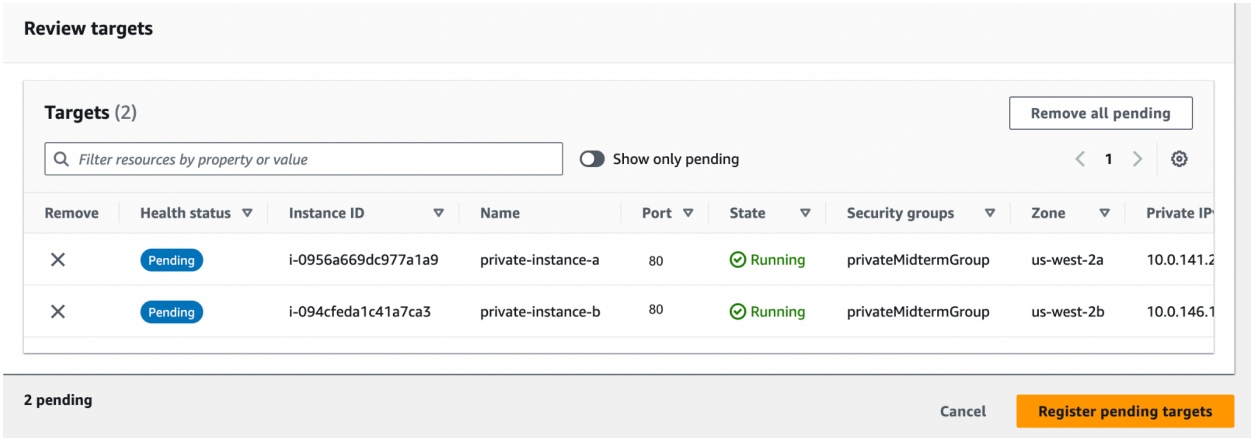


This will automatically add two new CNAME type records to Route 53, double check Route 53 to make sure they are there. Once the records have been created you will have to wait for the status of the certificate to change to “success”. You now have a SSL certificate.

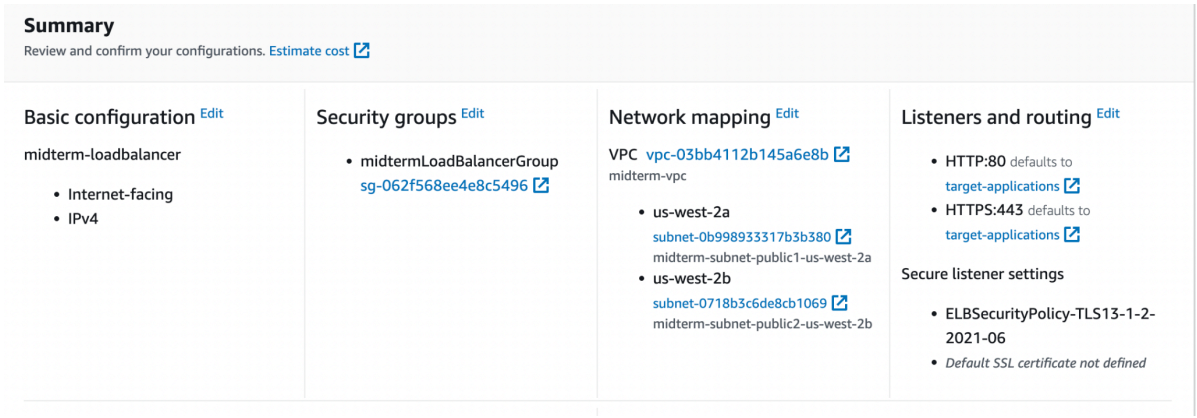
Load Balancer

The load balancer is what is going to make our deployment highly available and allow our clients to connect to our application through it.

Now that we have a SSL certificate we can create our Load Balancer. On AWS in the EC2 dashboard select “load balancer” on the left side. Then click “create load balancer”. There are three load balancer types, find Application Load Balancer scroll down and click “create”. Give the load balancer a name, make sure its scheme is internet-facing, and make the IP address type IPv4. Under the network mapping section put in our VPC and add the public subnet for each AZ. Make sure that you add the public subnet not the private subnet or else our clients will not be able to reach the load balancer. The security group will be the load balancer security group that we made earlier. Now go to the listeners and routing section and select “create target group”. Our target type will be instances. Give the target group a descriptive name. The protocol should be HTTP (port 80). Select IPv4 for IP address type, our VPC, and you can leave the protocol version as default (HTTP1). For the health check we can leave the protocol as HTTP and give a path that should return a 200 OK response when pinged, I will use “/health”. We can now select “next”. This should bring us to the available instances, there should be four of them. Select our two private instances, leave the port as 80, and click “include as pending below”:



Now we can click “register pending targets” to create our target group. Now navigate back to where we were creating the load balancer. In the listeners and routing section add our target group to the forwarding section of the HTTP listener. Then at the bottom click “add listener”; change the protocol of this listener to HTTPS and again add our target group to the forwarding section. Scroll down to the secure listener settings portion. Keep the security policy as what is recommended but add our SSL certificate to the default SSL/TLS certificate; it will be from ACM. Scroll past the optional sections and double check that the preview looks something along the lines of this:



Now select “create load balancer” and we have a functioning load balancer. Once we have our load balancer we can actually put our application on the private instances and we will be fully deployed. We also have to make sure that our application returns a 200 OK response at “/health” so the load balancer knows they are up. You can use whatever you would like to get your application code into the private instances, I would suggest SCP, but make sure you also install any dependencies.

Once your application has been deployed you should be able to put your domain into a web browser and see your new fully deployed application. In this tutorial you should have learned: how to create a VPC and why we need it, the appropriate way to set up security groups for a bastion host secured application, an easy way to launch your instances within the VPC, how to connect a domain to AWS, an easy way to get an SSL certificate connected to your application, and finally how to set up a load balancer to serve your clients. Thank you for following along :)