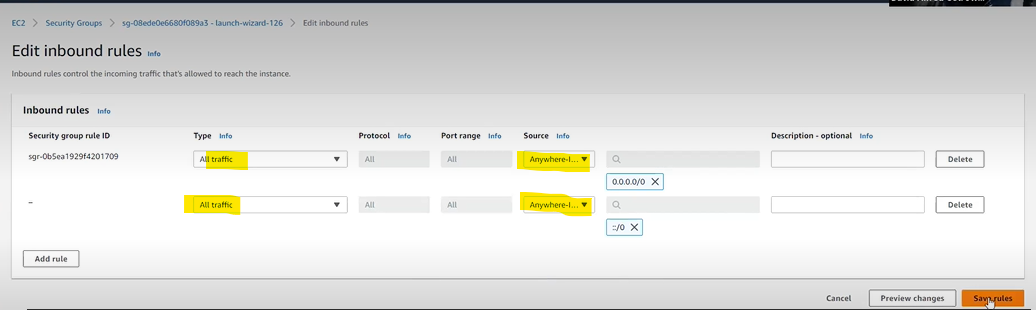
Deploying a Golang Web Server to Cloud

# AWS: Deployment to Elastic Beanstalk

Takes a Golang webserver & deploying it to Elastic Beanstalk as a service application (a place to deploy websites)

1. Create a new EC2 instance
2. Access instance using PuTTY
3. Make a new directory on instance

(Typically, you’d install Golang, but since we’ll do it in Docker, we don’t need to do it here)

1. Install Docker: “sudo snap install docker”
2. Install Golang: “vi main.go”
   1. “vi” = visual editor… it means you can edit the file
3. Set up a Docker file: “vi Dockerfile”
4. Build docker container: “sudo build docker -t golangdockertest .”
5. Now test out the web server: “sudo docker run -p 80:80 -tid golangdockertest”
6. Test it before deploying to Elastic Beanstalk
   1. Change traffic rules on instance
      1. Instance
      2. Security tab
      3. Security group (click link)
      4. Edit inbound rules
      5. Change type to All traffic
      6. Add traffic rules:
         1. 
   2. Grab the DNS from the instance
      1. Go to instance
      2. Details tab
      3. Copy public DNS
   3. Put DNS in browser, remember to change the port to 8080
7. Create a zip file with all of the files
   1. Putty: “sudo apt update zip” to get zipper application
   2. Putty: “zip files.zip \*”
8. Now just download the zip file, then upload it to the GUI (for now, the quick & dirty way of doing it)
9. Create IAM role
   1. On AWS, search IAM
   2. On left column, click Roles
   3. Create new
   4. Select AWS service, EC2 use case, click next
   5. Add permissions
      1. Search permission roles: AWSElasticBeanstalk\*
      2. Select these:
         1. AWSElasticBeanstockWebTier
         2. AWSElasticBeanstalkWorkingTier
         3. AWSElasticBeanstalkMultiContainerDocker
   6. Select next
   7. Name the role
10. Go to Elastic Beanstalk on AWS
11. Create Application (left-hand side menu)
    1. Step 1: Configure environment:
       1. Name it
       2. Create
       3. Create new environment
          1. Default = webserver environment
       4. Platform = Docker on Linux 2
       5. Upload code 🡪 the zip file you already saved
    2. Step 2: Configure service access
       1. Use existing service role
       2. Under EC2 instance profile, select the newly created IAM role
    3. Step 3: Set up networking, database, and tags
       1. Default settings
    4. Step 4: Configure instance traffic & scaling
       1. Default settings
    5. Step 5: Configure updates, monitoring, & logging
    6. Submit
12. Application will take a few minutes to spin up, but will generate a URL eventually

Delete applications you aren’t using or you’ll get charged for them.