**What is Docker?**

**How to create Docker file?**

**How to build Docker file?**

**How to register Docker hub?**

**How to run Docker file?**

**How to create ECR?**

**How to create ECS?**

**Lab work :**

**1.First install docker in amzon os server machine?**

**Syntax: yum install docker –y**

**2. start the service docker**

**Syntax: systemctl start docker**

**3. check the status docker is running are not**

**Syntax: systemctl status docker**

**4. I want to run one application like freecss template end point using the docker.**

**Dockerfile create start with capsletter if not give capslock it will not work dockerfile.**

**Lab work:**

**1.[root@ip-10-0-99-40 ~]# yum update & yum install docker –y**

**2.[root@ip-10-0-99-40 ~]# systemctl start docker**

**3.[root@ip-10-0-99-40 ~]# systemctl status docker**

**● docker.service - Docker Application Container Engine**

**Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)**

**Active: active (running) since Mon 2022-02-07 06:01:10 UTC; 10s ago**

**Docs: https://docs.docker.com**

**Process: 3478 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)**

**Process: 3477 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)**

**Main PID: 3481 (dockerd)**

**Tasks: 7**

**Memory: 37.4M**

**4.Now I’m downloading freecss template in amazon server? Using the wget**

[**https://www.free-css.com/assets/files/free-css-templates/download/page275/elegant.zip**](https://www.free-css.com/assets/files/free-css-templates/download/page275/elegant.zip) **this link i download**

**[root@ip-10-0-99-40 ~]# wget [root@ip-10-0-99-40 ~]# https://www.free-css.com/assets/files/free-css-templates/download/page275/elegant.zip**

**--2022-02-07 06:06:30-- http://%5Broot@ip-10-0-99-40/**

**Resolving ip-10-0-99-40 (ip-10-0-99-40)... 10.0.99.40**

**Connecting to ip-10-0-99-40 (ip-10-0-99-40)|10.0.99.40|:80... failed: Connection refused.**

**FINISHED --2022-02-07 06:06:32--**

**Total wall clock time: 1.6s**

**Downloaded: 1 files, 408K in 0.7s (552 KB/s)**

**[root@ip-10-0-99-40 ~]# -bash: https://www.free-css.com/assets/files/free-css-templates/download/page275/elegant.zip: No such file or directory**

**-bash: -bash:: command not found**

**[root@ip-10-0-99-40 ~]#**

**Now I’m to going to unzip the link file? Using the unzip.**

**[root@ip-10-0-99-40 ~]# unzip elegant.zip**

**[root@ip-10-0-99-40 ~]# ls**

**Elegant**

**[root@ip-10-0-99-40 ~]# mv Elegant elegant**

**[root@ip-10-0-99-40 ~]#**

**When you creating the Dockerfile it will not take the caps letter that wise I change the name ? now I creating the docker file and register the docker hub.**

**First I go to login Docker hub then I will search the httpd . here I will take the**

**Create a Dockerfile in your project**

**FROM httpd:2.4**

**COPY ./public-html/ /usr/local/apache2/htdocs/ so copy this one go to your**

**Docker machine / vi Dockerfile / paste here.**

**[root@ip-10-0-99-40 ~]# ls**

**elegant**

**[root@ip-10-0-99-40 ~]#**

**[root@ip-10-0-99-40 ~]# ls**

**elegant**

**[root@ip-10-0-99-40 ~]# vi Dockerfile**

**FROM httpd:2.4**

**COPY elegant /usr/local/apache2/htdocs/**

**:wq!**

**[root@ip-10-0-99-40 ~]# ls**

**Dockerfile elegant**

**[root@ip-10-0-99-40 ~]# docker build -t darkdocker1/elegant:1 .**

**Sending build context to Docker daemon 591.4kB**

**Step 1/2 : FROM httpd:2.4**

**2.4: Pulling from library/httpd**

**5eb5b503b376: Pull complete**

**a43a76ccc967: Pull complete**

**942bd346e7f7: Pull complete**

**cdb155854ae6: Pull complete**

**10c4d45228bf: Pull complete**

**Digest: sha256:5cc947a200524a822883dc6ce6456d852d7c5629ab177dfbf7e38c1b4a647705**

**Status: Downloaded newer image for httpd:2.4**

**---> a8ea074f4566**

**Step 2/2 : COPY elegant /usr/local/apache2/htdocs/**

**---> 80293e049685**

**Successfully built 80293e049685**

**Successfully tagged darkdocker1/elegant:1**

**[root@ip-10-0-99-40 ~]# docker login -u darkdocker1**

**Password:**

**WARNING! Your password will be stored unencrypted in /root/.docker/config.json.**

**Configure a credential helper to remove this warning. See**

**https://docs.docker.com/engine/reference/commandline/login/#credentials-store**

**Login Succeeded**

**[root@ip-10-0-99-40 ~]# docker push darkdocker1/elegant:1**

**The push refers to repository [docker.io/darkdocker1/elegant]**

**61ad85fb6052: Pushed**

**86125726a640: Mounted from library/httpd**

**44e3d894866b: Mounted from library/httpd**

**45a719f83e9c: Mounted from library/httpd**

**3ce7912af936: Mounted from library/httpd**

**7d0ebbe3f5d2: Mounted from library/httpd**

**1: digest: sha256:2958645db27809a35e5b89d3d4104ffaa51730e26e434238f1da49a26e5cf837 size: 1575**

**[root@ip-10-0-99-40 ~]# docker images**

**REPOSITORY TAG IMAGE ID CREATED SIZE**

**darkdocker1/elegant 1 80293e049685 4 minutes ago 144MB**

**httpd 2.4 a8ea074f4566 11 days ago 144MB**

**[root@ip-10-0-99-40 ~]# docker run -d -p 80:80 80293e049685**

**407aa68962ed21736886851b0e7dc9ca4f7d94faf326d18049020ca5c11363de**

**[root@ip-10-0-99-40 ~]#**

**[root@ip-10-0-99-40 ~]# docker ps -a**

**CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES**

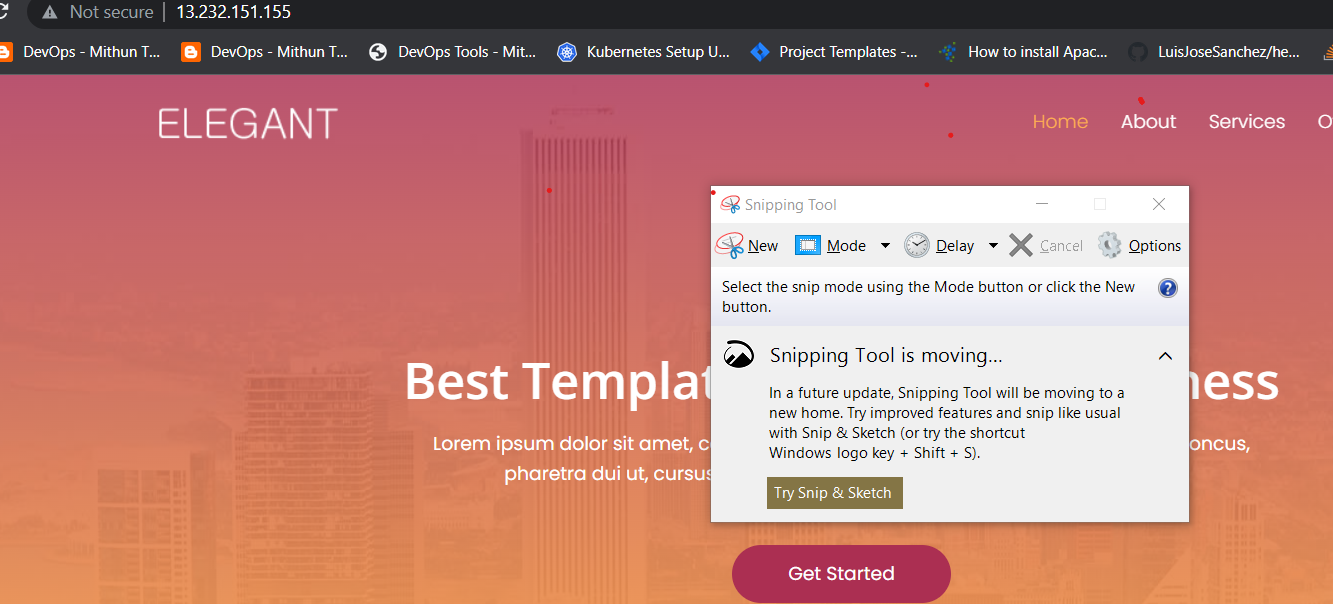
**407aa68962ed 80293e049685 "httpd-foreground" About a minute ago Up About a minute 0.0.0.0:80->80/tcp, :::80->80/tcp boring\_murdock**

**[root@ip-10-0-99-40 ~]#**

**Now I’m taking the end ip go to the browser paste here it will display what ever**

**Docker build images that images it will display outside that end user it see**

**Actual developer create web application the application devOps team we build the docker images give to the end point to access the browser.**

****

**Now I’m going to creating ECR in aws?**

**Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable. ... This is so that specified users or Amazon EC2 instances can access your container repositories and images.**

**Prerequisite:**

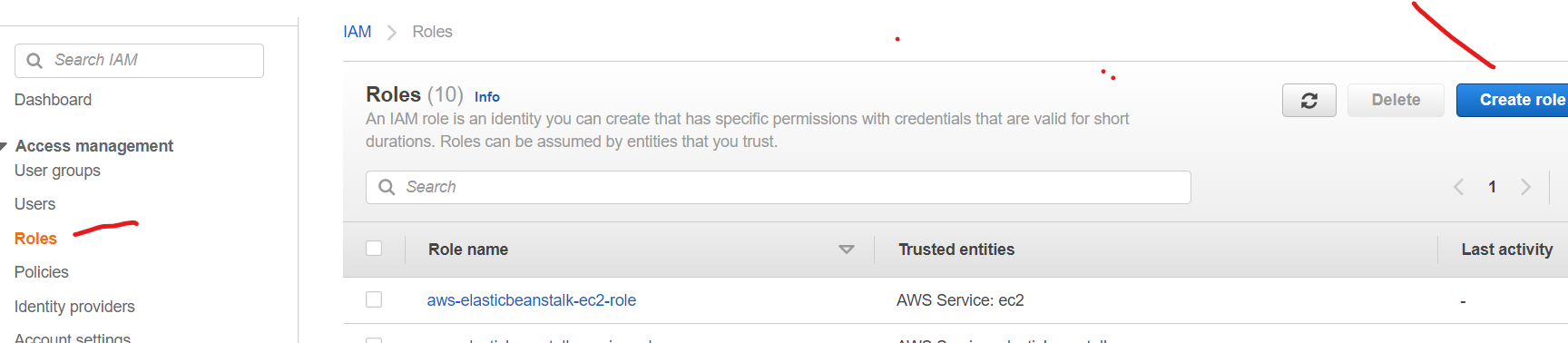
**Creating IAM role and giving the permissions.**

**Creating the ECR to Pushing the docker Images.**

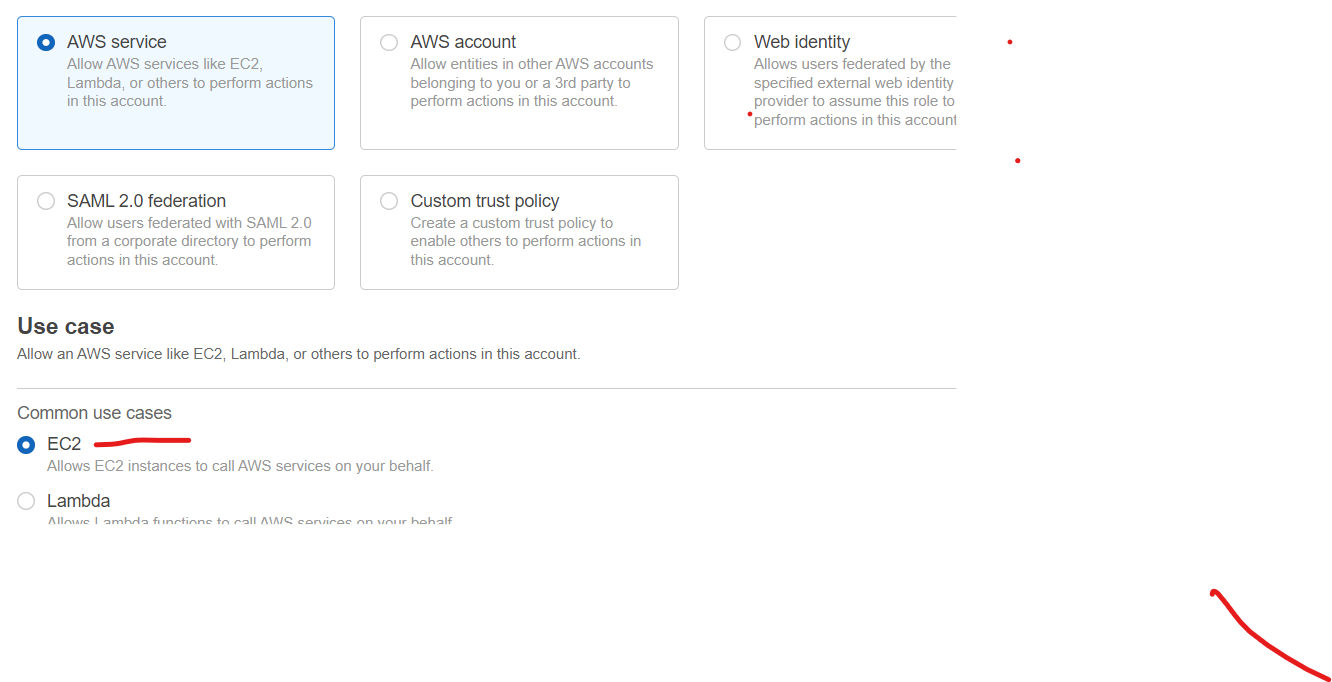
**IAM role giving the permissions**

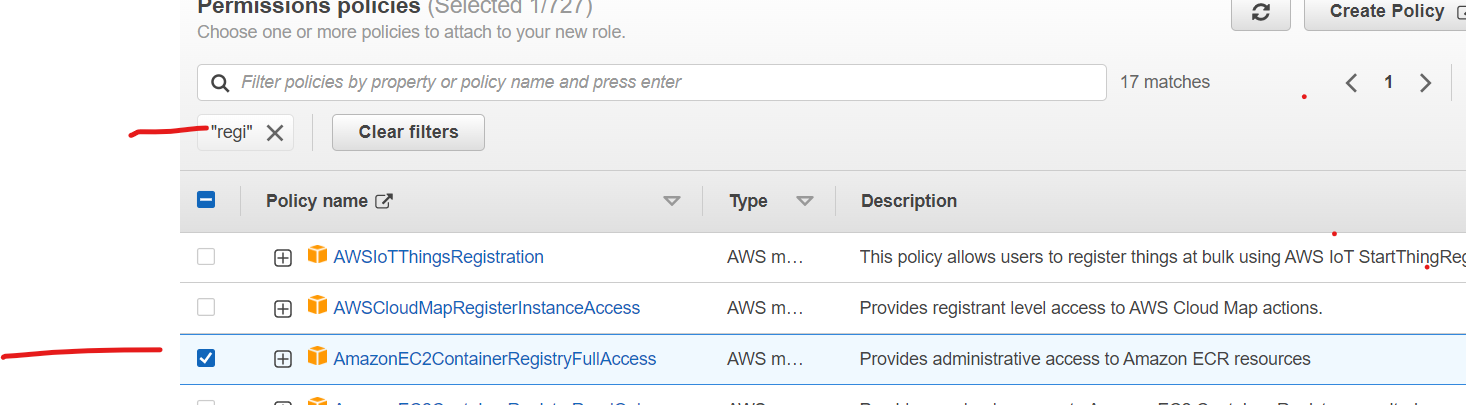
**AWS Identity and Access Management (IAM) provides fine-grained access control across all of AWS. ... With IAM policies, you manage permissions to your workforce and systems to ensure least-privilege permissions.**

**1.Creating the I’m role giving ec2 the permissions**

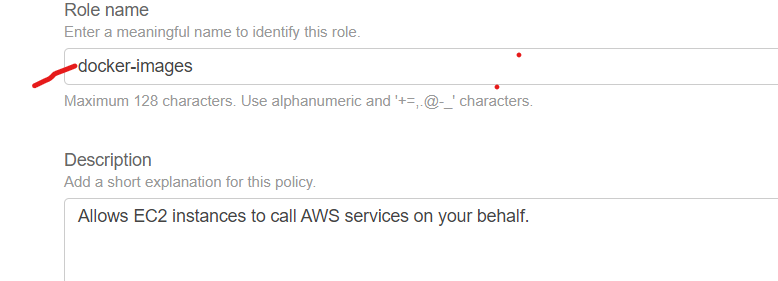
****

**2.Click the create role and select the EC2 click the Next**

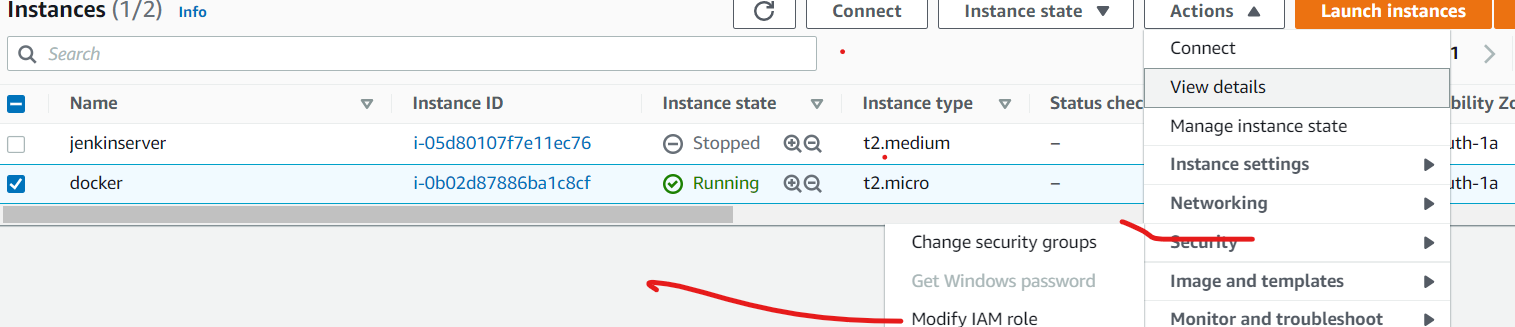
**click the Next**

**3.Give the permission that ec2 instance&AmazonEc2contaierRegistryfullaccess**

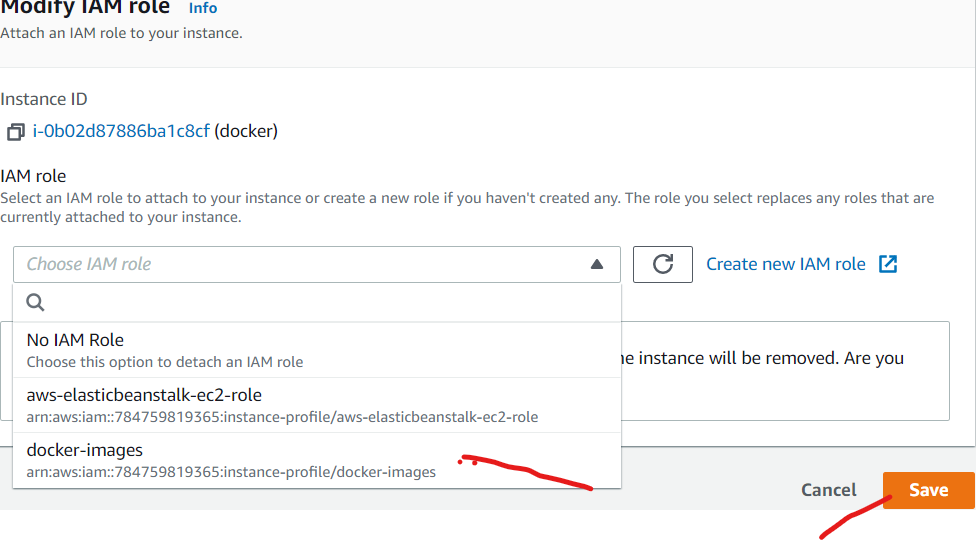
**4.Give the Role name and click the create role**

****

**5.And go the ec2 machine click the action button click the security click the modify IAM role.**

****

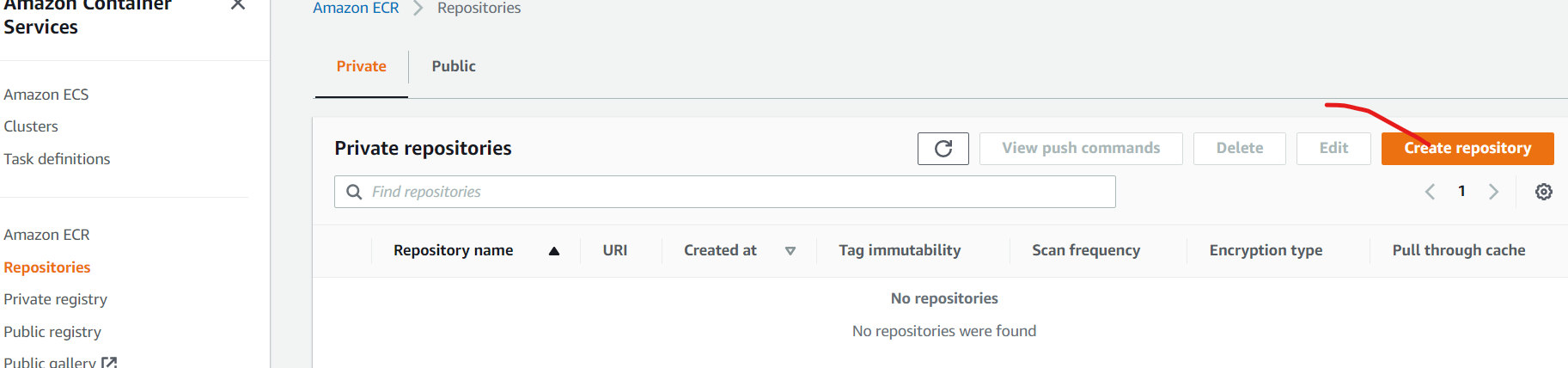
**6.Select the docker –images and save it**

****

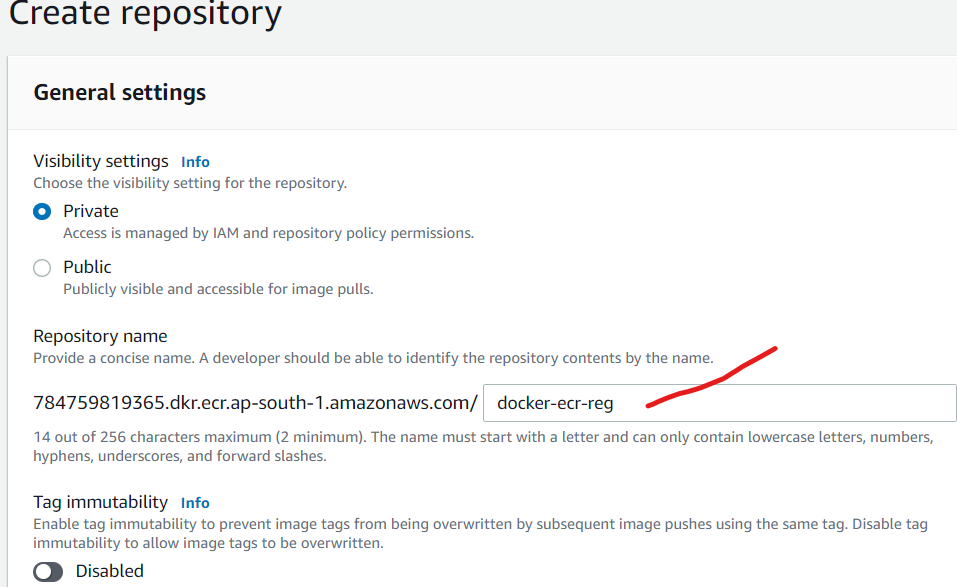
**ECR Amazon Elastic Container Registry**

**Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable. Amazon ECR supports private repositories with resource-based permissions using AWS IAM. Go to the AWS all services search ECR it will display the just click on it wil open the.**

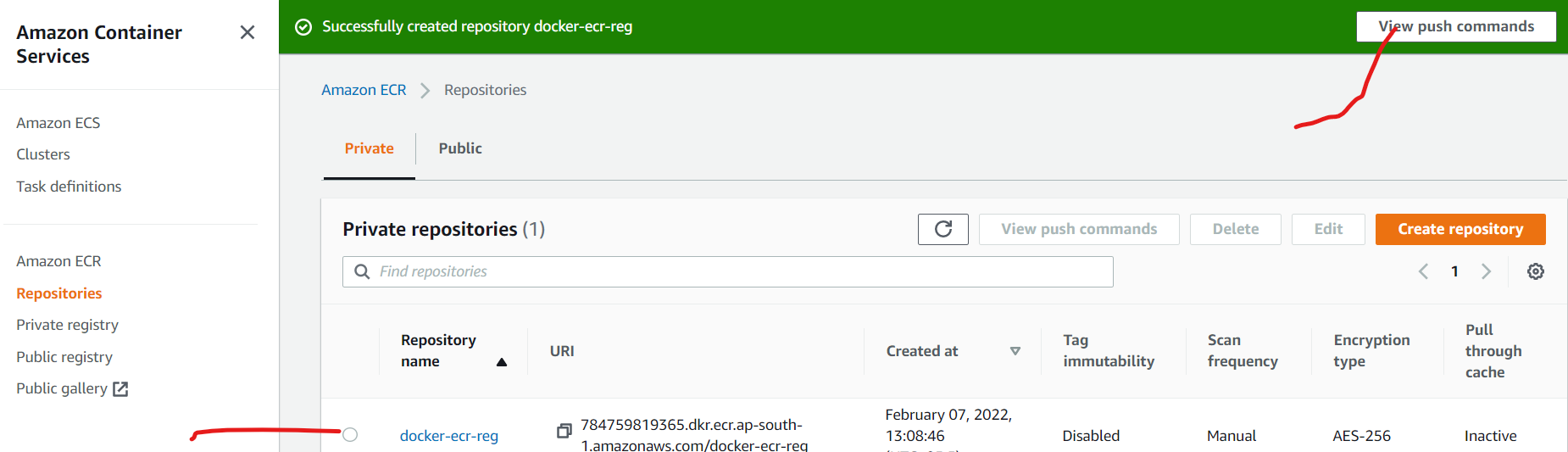
1. **Page and create the ECR**

****

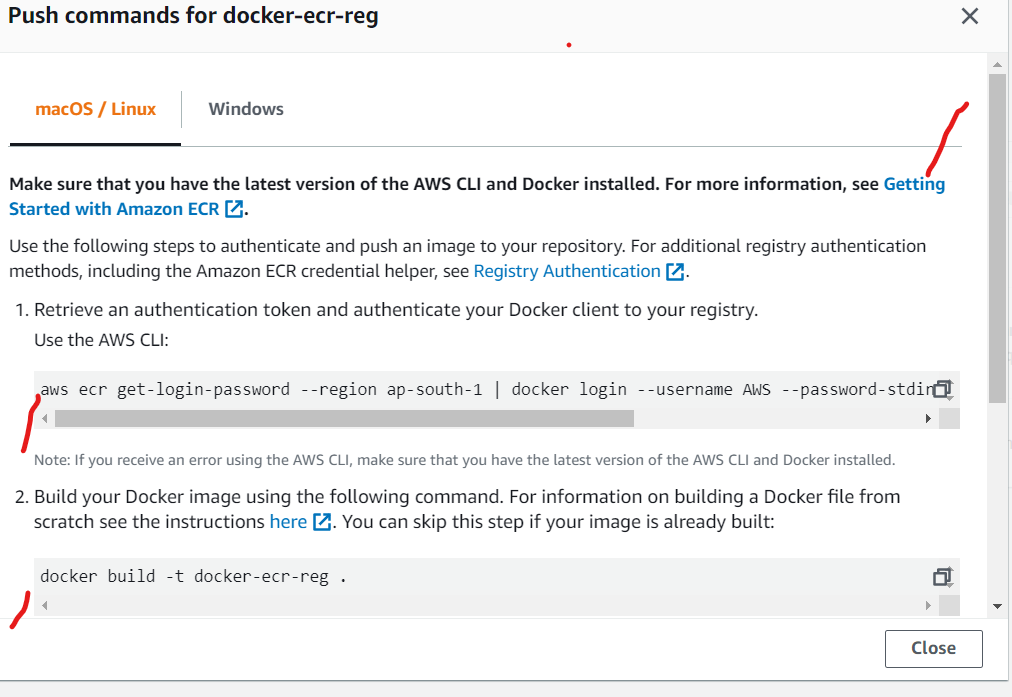
1. **Here we enter name docker-ecr-reg and don’t change anything just click one create repository**

****

**Once we create successfully docker-ecr-reg here we can observer green color just click the view push commands it display macos/linux | windows.**

****

**Here we see the Getting started with Amazon ECR we need follow below steps to register docker image& Getting first right click and open new tab the.**

****

**1.step sudo yum update -y**

**2.step sudo amazon-linux-extras install docker**

**3.step sudo service docker start**

**4.step docker info**

**5.step aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 784759819365.dkr.ecr.ap-south-1.amazonaws.com**

**6.step docker build -t docker-ecr-reg .**

**7.step docker tag docker-ecr-reg:latest 784759819365.dkr.ecr.ap-south-1.amazonaws.com/docker-ecr-reg:latest**

**8.step docker push 784759819365.dkr.ecr.ap-south-1.amazonaws.com/docker-ecr-reg:latest**

**[root@ip-10-0-99-40 ~]# sudo yum update -y**

**Loaded plugins: extras\_suggestions, langpacks, priorities, update-motd**

**amzn2-core | 3.7 kB 00:00**

**No packages marked for update**

**[root@ip-10-0-99-40 ~]# sudo amazon-linux-extras install docker**

**Installing docker**

**Loaded plugins: extras\_suggestions, langpacks, priorities, update-motd**

**Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10**

**17 metadata files removed**

**6 sqlite files removed**

**0 metadata files removed**

**Loaded plugins: extras\_suggestions, langpacks, priorities, update-motd**

**amzn2-core | 3.7 kB 00:00**

**amzn2extra-docker | 3.0 kB 00:00**

**amzn2extra-kernel-5.10 | 3.0 kB 00:00**

**(1/7): amzn2-core/2/x86\_64/group\_gz | 2.5 kB 00:00**

**(2/7): amzn2-core/2/x86\_64/updateinfo | 442 kB 00:00**

**(3/7): amzn2extra-kernel-5.10/2/x86\_64/primary\_db | 6.4 MB 00:00**

**60 mock2 available [ =stable ]**

**61 dnsmasq2.85 available [ =stable ]**

**[root@ip-10-0-99-40 ~]# sudo service docker start**

**Redirecting to /bin/systemctl start docker.service**

**[root@ip-10-0-99-40 ~]# sudo usermod -a -G docker ec2-user**

**[root@ip-10-0-99-40 ~]# docker info**

**Client:**

**Context: default**

**Debug Mode: false**

**Server:**

**Containers: 1**

**Running: 1**

**Paused: 0**

**Stopped: 0**

**Images: 2**

**Server Version: 20.10.7**

**Storage Driver: overlay2**

**Backing Filesystem: xfs**

**Supports d\_type: true**

**Native Overlay Diff: true**

**userxattr: false**

**Logging Driver: json-file**

**Cgroup Driver: cgroupfs**

**Cgroup Version: 1**

**Plugins:**

**Volume: local**

**Network: bridge host ipvlan macvlan null overlay**

**Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog**

**Swarm: inactive**

**Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc**

**Default Runtime: runc**

**Init Binary: docker-init**

**containerd version: d71fcd7d8303cbf684402823e425e9dd2e99285d**

**runc version: 84113eef6fc27af1b01b3181f31bbaf708715301**

**init version: de40ad0**

**Security Options:**

**seccomp**

**Profile: default**

**Kernel Version: 5.10.93-87.444.amzn2.x86\_64**

**Operating System: Amazon Linux 2**

**OSType: linux**

**Architecture: x86\_64**

**CPUs: 1**

**Total Memory: 965.5MiB**

**Name: ip-10-0-99-40.ap-south-1.compute.internal**

**ID: QG3I:PJ5V:7PB7:PGIS:T5CJ:WWJS:6KYI:X3UH:UVMJ:I63P:4ZU6:O7IF**

**Docker Root Dir: /var/lib/docker**

**Debug Mode: false**

**Username: darkdocker1**

**Registry: https://index.docker.io/v1/**

**Labels:**

**Experimental: false**

**Insecure Registries:**

**127.0.0.0/8**

**Live Restore Enabled: false**

**[root@ip-10-0-99-40 ~]# aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 784759819365.dkr.ecr.ap-south-1.amazonaws.com**

**WARNING! Your password will be stored unencrypted in /root/.docker/config.json.**

**Configure a credential helper to remove this warning. See**

**https://docs.docker.com/engine/reference/commandline/login/#credentials-store**

**Login Succeeded**

**[root@ip-10-0-99-40 ~]# docker build -t docker-ecr-reg .**

**Sending build context to Docker daemon 596.5kB**

**Step 1/2 : FROM httpd:2.4**

**---> a8ea074f4566**

**Step 2/2 : COPY elegant /usr/local/apache2/htdocs/**

**---> Using cache**

**---> 80293e049685**

**Successfully built 80293e049685**

**Successfully tagged docker-ecr-reg:latest**

**[root@ip-10-0-99-40 ~]# docker tag docker-ecr-reg:latest 784759819365.dkr.ecr.ap-south-1.amazonaws.com/docker-ecr-reg:latest**

**[root@ip-10-0-99-40 ~]# docker push 784759819365.dkr.ecr.ap-south-1.amazonaws.com/docker-ecr-reg:latest**

**The push refers to repository [784759819365.dkr.ecr.ap-south-1.amazonaws.com/docker-ecr-reg]**

**61ad85fb6052: Pushed**

**86125726a640: Pushed**

**44e3d894866b: Pushed**

**45a719f83e9c: Pushed**

**3ce7912af936: Pushed**

**7d0ebbe3f5d2: Pushed**

**latest: digest: sha256:2958645db27809a35e5b89d3d4104ffaa51730e26e434238f1da49a26e5cf837 size: 1575**

**[root@ip-10-0-99-40 ~]# docker images**

**REPOSITORY TAG IMAGE ID CREATED SIZE**

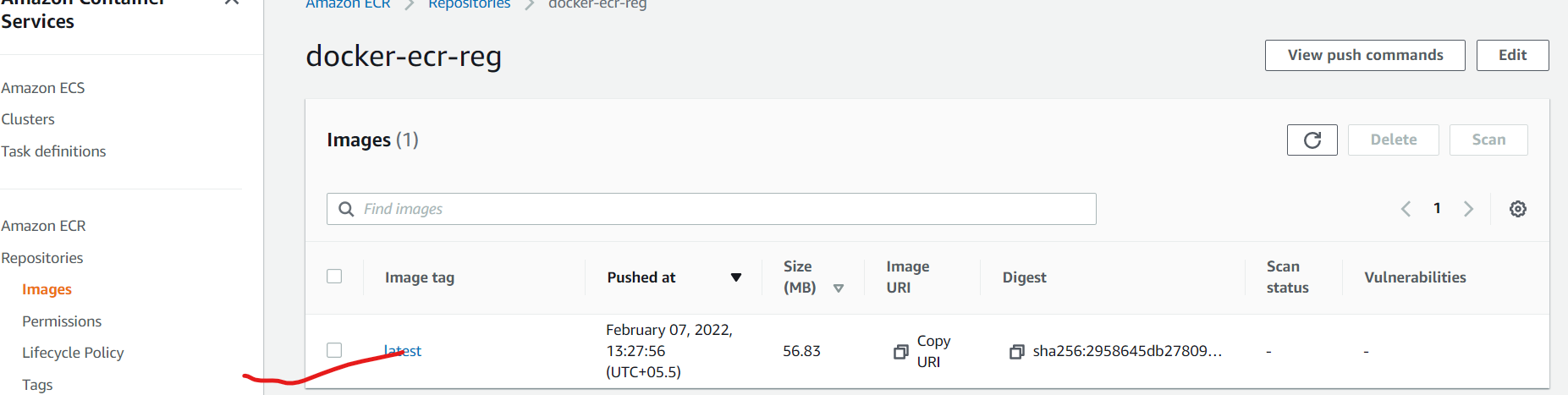
**784759819365.dkr.ecr.ap-south-1.amazonaws.com/docker-ecr-reg latest 80293e049685 2 hours ago 144MB**

**darkdocker1/elegant 1 80293e049685 2 hours ago 144MB**

**docker-ecr-reg latest 80293e049685 2 hours ago 144MB**

**httpd 2.4 a8ea074f4566 11 days ago 144MB**

**Now see can docker images created in ECR.**

****

**This way we creating the ECR steps**

**How to delete docker images and container**

1. **docker build –rm .**
2. **du –sh /var/lib/docker**
3. **df –h /var/lib/docker**
4. **docker run –i -t**
5. **docker ps -a**
6. **docker rm container can’t remove one that is running**
7. **docker stop container**
8. **docker rm container**
9. **docker rm container container container**
10. **docker rmi image**
11. **docker rmi image image image image**
12. **docker rmi $(docker images –a –q) Removes all images**
13. **docker rm $(docker ps –a –q) Removes all containers**

**ECS Amazon Elastic Container Service**

**Amazon Elastic Container Service (ECS) is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances.**

**Few of points:**

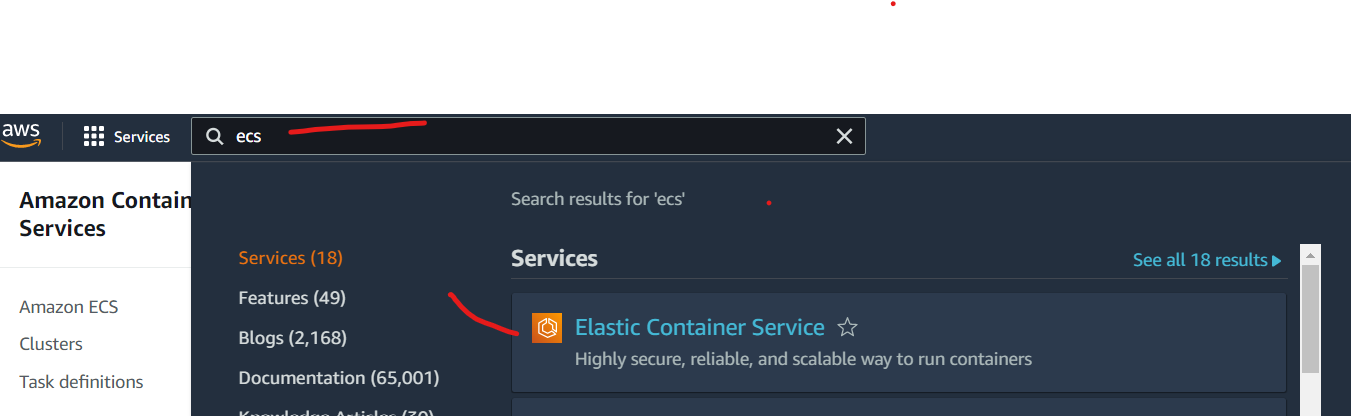
**1. Amazon Elastic Container Service (Amazon ECS) is a very scalable, durable container management service that makes it simple to run, halt, and control containers on a cluster.**

**2.The containers are specified in a task definition that handles to run individual tasks or tasks within a service.**

**3.Amazon ECS allows you to launch and hold container-based applications by using simple API calls. You can also recover the state of your cluster from a centralized service and have access to several familiar Amazon EC2 features.**

**Steps: labwork**

**Frist go the AWS all services search the ECS**

****