Amazon Elastic Container Registry (ECR)

1. What Is Amazon Elastic Container Registry?

- The most commonly adopted way to store and deliver **Docker images** is through **Docker Registry**, an open source application by Docker that hosts **Docker repositories**. This application can be deployed on-premises, as well as used as a service from multiple providers, such as **Docker Hub**, **Quay.io**, and **AWS ECR**.

- **ECR** is AWS's approach to a hosted Docker registry, where there's one registry per account. It uses AWS IAM to authenticate and authorize users to push and pull images by using **Docker CLI**. By default, the limits for both repositories and images are set to **1,000**.

- ECR stores your container images in Amazon S3.

- ECR supports Docker Registry HTTP API V2 allowing you to use Docker CLI commands or your preferred Docker tools in maintaining your existing development workflow.

2. Components of Amazon ECR

- **Registry**:
  - You can create image repositories in your registry and store images in them.
  - The URL for your default registry is

    `https://`*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com`

- **Authentication Token**: Your Docker client must authenticate to Amazon ECR registries as an AWS user before it can push and pull images. The AWS CLI **get-login** command provides you with authentication credentials to pass to Docker.

- **Repository**: An Amazon ECR image repository contains your Docker images.

- **Repository policy**:
  - You can control access to your repositories and the images within them with repository policies.
  - **ECR lifecycle policies** enable you to specify the lifecycle management of images in a repository.

- **Image**: You can push and pull Docker images to your repositories. You can use these images locally on your development system, or you can use them in Amazon ECS task definitions.

2.1 How to Get Started with Amazon ECR

- To use **Amazon ECR**, you must be set up to install the **AWS Command Line Interface** (**CLI**) and **Docker**.

3. Docker Basics for Amazon ECR

Step-1: Installing Docker on Amazon Linux 2

1. Launch an instance with the Amazon Linux 2 AMI.
2. Connect to your instance.
3. Update the installed packages and package cache on your instance.
   ```
   $ sudo yum update –y
   ```
4. Install the most recent Docker Community Edition package.
   ```
   $ sudo amazon-linux-extras install docker
   ```
5. Start the Docker service.
   ```
   $ sudo service docker start
   ```
6. Add the `ec2-user` to the `docker` group so you can execute Docker commands without using `sudo`.
   ```
   $ sudo usermod -a -G docker ec2-user
   ```
7. Log out and log back in again to pick up the new docker group permissions.
8. Verify that the `ec2-user` can run Docker commands without `sudo`.
   ```
   $ docker info
   ```

Step-2: Create a Docker Image

You create a Docker image of a simple web application, and test it on your local system or EC2 instance, and then push the image to a container registry (such as Amazon ECR or Docker Hub) so you can use it in an ECS task definition.

1. Create a file called Dockerfile. A Dockerfile is a manifest that describes the base image to use for your Docker image and what you want installed and running on it.
   ```
   $ touch Dockerfile
   ```
2. Edit the `Dockerfile` you just created and add the following content.
   ```
   FROM ubuntu:18.04
   # Install dependencies

   RUN apt-get update && \
    apt-get -y install apache2

   # Install apache and write hello world message
   ```

```
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo '. /etc/apache2/envvars' > /root/run_apache.sh && \
 echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh && \
 echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

3. Build the Docker image from your Dockerfile.

```
$ docker build -t hello-world .
```

4. Run **docker images** to verify that the image was created correctly.

```
$ docker images --filter reference=hello-world
```

Output:

```
REPOSITORY        TAG         IMAGE ID          CREATED             SIZE
hello-world      latest    e9ffedc8c286       4 minutes ago       241MB
```

5. Run the newly built image. The `-p 80:80` option maps the exposed port 80 on the container to port 80 on the host system.

```
$ docker run -t -i -p 80:80 hello-world
```

6. Open a browser and point to the server that is running Docker and hosting your container.
7. Stop the Docker container by typing **Ctrl + c**.

Step-3: Push your image to Amazon Elastic Container Registry

Amazon ECR is a managed AWS Docker registry service. Customers can use the familiar Docker CLI to push, pull, and manage images.

1. Create an Amazon ECR repository to store your `hello-world` image. Note the `repositoryUri` in the output.

```
$ aws ecr create-repository --repository-name hello-repository --region region
```

Output:

```
{
    "repository": {
        "registryId": "aws_account_id",
        "repositoryName": "hello-repository",
        "repositoryArn":
"arn:aws:ecr:region:aws_account_id:repository/hello-repository",
        "createdAt": 1505337806.0,
        "repositoryUri":
"aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository"
    }
}
```

2. Tag the `hello-world` image with the `repositoryUri` value from the previous step.

   ```
   $ docker tag hello-world aws_account_id.dkr.ecr.region.amazonaws.com/hello-
   repository
   ```

3. Run the **aws ecr get-login --no-include-email** command to get the **docker login** authentication command string for your registry.

   ```
   $ aws ecr get-login --no-include-email --region region
   ```

4. Run the **docker login** command that was returned in the previous step. This command provides an authorization token that is valid for 12 hours.

NOTE: When you execute this **docker login** command, the command string can be visible to other users on your system in a process list (**ps -e**) display. Because the **docker login** command contains authentication credentials, there is a risk that other users on your system could view them this way. They could use the credentials to gain push and pull access to your repositories. If you are not on a secure system, you should consider this risk and log in interactively by omitting the `-p` *`password`* option, and then entering the password when prompted.

5. Push the image to Amazon ECR with the `repositoryUri` value from the earlier step.

   ```
   $ docker push aws_account_id.dkr.ecr.region.amazonaws.com/hello-
   repository
   ```

Step-4: Clean up

When you are done experimenting with your Amazon ECR image, you can delete the repository so you are not charged for image storage.

```
$ aws ecr delete-repository --repository-name hello-repository --region
region --force
```

4. Amazon ECR Registries

- Amazon ECR registries to host your images in a highly available and scalable architecture, allowing you to deploy containers reliably for your applications.

- Your account has read and write access to the repositories in your default registry.

- IAM users require permissions to make calls to the Amazon ECR APIs and to push or pull images from your repositories.

- You must authenticate your Docker client to a registry so that you can use the **docker push** and **docker pull** commands to push and pull images to and from the repositories in that registry.

- Repositories can be controlled with both IAM user access policies and repository policies.

4.1 Registry Authentication

- The Docker CLI does not support native IAM authentication methods. Additional steps must be taken so that Amazon ECR can authenticate and authorize Docker push and pull requests.

- You must use the `GetAuthorizationToken` API operation to retrieve a base64-encoded authorization token containing the username `AWS` and an encoded password.

- The AWS CLI `get-login` command simplifies this by retrieving, decoding, and converting the authorization token into a pre-generated `docker login` command.

To authenticate Docker to an Amazon ECR registry with get-login

- Run the `aws ecr get-login` command. The example below is for the default registry associated with the account making the request. To access other account registries, use the `--registry-ids` *`aws_account_id`* option.

  ```
  $ aws ecr get-login --region region --no-include-email
  ```

  Output:

  ```
  docker login -u AWS -p password https://aws_account_id.dkr.ecr.us-
  east-1.amazonaws.com
  ```

- The resulting output is a **docker login** command that you use to authenticate your Docker client to your Amazon ECR registry.

- Copy and paste the `docker login` command into a terminal to authenticate your Docker CLI to the registry. This command provides an authorization token that is valid for the specified registry for 12 hours.

4.2 HTTP API Authentication

- Amazon ECR supports the Docker Registry HTTP API, because Amazon ECR is a private registry,

- You must provide an authorization token with every HTTP request.

- You can add an HTTP authorization header using the `-H` option for **curl** and pass the authorization token provided by the **get-authorization-token** AWS CLI command.

To authenticate with the Amazon ECR HTTP API

- Retrieve an authorization token with the AWS CLI and set it to an environment variable.

  ```
  TOKEN=$(aws ecr get-authorization-token --output text --query
  'authorizationData[].authorizationToken')
  ```

- To authenticate to the API, pass the `$TOKEN` variable to the `-H` option of **curl**. For example, the following command lists the image tags in an Amazon ECR repository.

```
curl -i -H "Authorization: Basic $TOKEN"
https://012345678910.dkr.ecr.us-east-
1.amazonaws.com/v2/amazonlinux/tags/list
```

Output:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Thu, 04 Jan 2018 16:06:59 GMT
Docker-Distribution-Api-Version: registry/2.0
Content-Length: 50
Connection: keep-alive



{"name":"amazonlinux","tags":["2017.09","latest"]}
```

5. Amazon ECR Repositories

- Amazon Elastic Container Registry (Amazon ECR) provides API operations to create, monitor, and delete image repositories and set permissions that control who can access them.

- Amazon ECR also integrates with the Docker CLI allowing you to push and pull images from your development environments to your repositories.

- Repository names can support namespaces, which you can use to group similar repositories.

- Example: there are several teams using the same registry, Team A could use the `team-a` namespace while Team B uses the `team-b` namespace. Each team could have their own image called `web-app`, but because they are each prefaced with the team namespace, the two images can be used simultaneously without interference. Team A's image would be called `team-a/web-app`, while Team B's image would be called `team-b/web-app`.

5.1 Creating a Repository

1. Open the **Amazon ECR console** at https://console.aws.amazon.com/ecr/repositories.
2. From the navigation bar, **choose the Region** to create your repository in.
3. In the navigation pane, choose **Repositories**.
4. On the **Repositories** page, choose **Create repository**.
5. For **Repository configuration**, enter a unique name for your repository.
6. For **Image tag mutability**, choose the tag mutability setting for the repository. Repositories configured with immutable tags will prevent image tags from being overwritten.

7. For **Image scanning**, choose the image scanning setting for the repository. Repositories configured to scan on push will start an image scan whenever an image is pushed, otherwise image scans need to be started manually.
8. Choose **Create repository**.
9. (Optional) Select the repository you created and choose **View push commands** to view the steps to push an image to your new repository.
   a. Retrieve the **docker login** command that you can use to authenticate your Docker client to your registry by pasting the **aws ecr get-login** command from the console into a terminal window.
   b. Run the **docker login** command that was returned in the previous step. This command provides an authorization token that is valid for 12 hours.
   c. (Optional) If you have a Dockerfile for the image to push, build the image and tag it for your new repository. Pasting the **docker build** command from the console into a terminal window. Make sure that you are in the same directory as your Dockerfile.
   d. Tag the image for your ECR registry and your new repository by pasting the **docker tag** command from the console into a terminal window. The console command assumes that your image was built from a Dockerfile in the previous step. If you did not build your image from a Dockerfile, replace the first instance of `*repository*:latest` with the image ID or image name of your local image to push.
   e. Push the newly tagged image to your ECR repository by pasting the **docker push** command into a terminal window.
   f. Choose **Close**.

- After you have created a repository, you can view its information in the AWS Management Console:
  - Which images are stored in a repository
  - Whether an image is tagged
  - The tags for the image
  - The SHA digest for the images
  - The size of the images in MiB
  - When the image was pushed to the repository

To view repository information

1. In the navigation pane, choose **Repositories**.
2. On the **Repositories** page, choose the repository to view.
3. On the **Repositories : *repository_name*** page, use the navigation bar to view information about an image.

- Choose **Images** to view information about the images in the repository. If there are untagged images that you would like to delete, you can select the box to the left of the repositories to delete and choose **Delete**.
- Choose **Permissions** to view the repository policies that are applied to the repository.
- Choose **Lifecycle Policy** to view the lifecycle policy rules that are applied to the repository. The lifecycle events history is also viewed here.
- Choose **Tags** to view the metadata tags that are applied to the repository.

5.2 Amazon ECR Repository Policy Examples

➢ **Allow other account**:

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AllowPull",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:user/pull-user-1",
          "arn:aws:iam::123456789012:user/pull-user-2"
        ]
      },
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    },
    {
      "Sid": "AllowAll",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/admin-user"
      },
      "Action": [
        "ecr:*"
      ]
    }
  ]
}
```

➢ **Restricting Access to Specific IP Addresses**

```
{
  "Version": "2012-10-17",
  "Id": "ECRPolicyId1",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "ecr:*",
      "Resource":  "arn:aws:ecr:us-east-1:123456789012:repository/my-
repo",
      "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "54.240.143.188/32"
                },
                "IpAddress": {
                    "aws:SourceIp": "54.240.143.0/24"
                }
       }
    }
  ]
}
```

5.3 Tagging an Amazon ECR Repository

- To help you manage your Amazon ECR repositories, you can optionally assign your own metadata to each repository in the form of *tags*.

- Tag an existing repository with multiple tags

```
aws ecr tag-resource --resource-arn
arn:aws:ecr:region:account_id:repository/repository_name --tags
Key=key1,Value=value1 key=key2,value=value2 key=key3,value=value3
```

- Untag an existing repository

```
aws ecr untag-resource --resource-arn
arn:aws:ecr:region:account_id:repository/repository_name --tag-keys
tag_key
```

- List tags for a repository

```
aws ecr list-tags-for-resource --resource-arn
arn:aws:ecr:region:account_id:repository/repository_name
```

- Create a repository and apply a tag

```
aws    ecr    create-repository    --repository-name    test-repo    --tags
Key=team,Value=devs
```

6. Images

- Amazon Elastic Container Registry (Amazon ECR) stores Docker images in image repositories. You can use the Docker CLI to push and pull images from your repositories.

- Amazon ECR requires that users have allow permissions to the `ecr:GetAuthorizationToken` API through an IAM policy before they can authenticate to a registry and push or pull any images from any Amazon ECR repository.

6.1 To push a Docker image to an Amazon ECR repository

If you have a Docker image available in your development environment, you can push it to an Amazon ECR repository with the **docker push** command.

1. Authenticate your Docker client to the Amazon ECR registry to which you intend to push your image. Authentication tokens must be obtained for each registry used, and the tokens are valid for 12 hours.
2. If your image repository does not exist in the registry you intend to push to yet, create it.
3. Identify the image to push. Run the **docker images** command to list the images on your system.

   ```
   $ docker images
   ```

   You can identify an image with the `repository:tag` value or the image ID in the resulting command output.

4. Tag your image with the Amazon ECR registry, repository, and optional image tag name combination to use. The registry format is `aws_account_id.dkr.ecr.region.amazonaws.com`. The repository name should match the repository that you created for your image. If you omit the image tag, we assume that the tag is `latest`.
5. The following example tags an image with the ID `e9ae3c220b23` as `aws_account_id.dkr.ecr.region.amazonaws.com/my-web-app`

   ```
   $ docker tag e9ae3c220b23
   aws_account_id.dkr.ecr.region.amazonaws.com/my-web-app
   ```

6. Push the image using the **docker push** command:

   ```
   $ docker push aws_account_id.dkr.ecr.region.amazonaws.com/my-web-app
   ```

7. (Optional) Apply any additional tags to your image and push those tags to Amazon ECR by repeating Step 4 and Step 5.
8. You can apply up to 100 tags per image in Amazon ECR.

6.2 Deleting an Image

To delete an image with the AWS Management Console

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories.
2. From the navigation bar, choose the Region that contains the image to delete.
3. In the navigation pane, choose **Repositories**.
4. On the **Repositories** page, choose the repository that contains the image to delete.
5. On the **Repositories:** ***`repository_name`*** page, select the box to the left of the image to delete and choose **Delete**.
6. In the **Delete image(s)** dialog box, verify that the selected images should be deleted and choose **Delete**.

To delete an image with the AWS CLI

1. List the images in your repository so that you can identify them by image tag or digest.

```
$ aws ecr list-images --repository-name my-repo
```

2. (Optional) Delete any unwanted tags for the image by specifying the tag of the image you want to delete.

```
$ aws ecr batch-delete-image --repository-name my-repo --image-ids
imageTag=latest
```

3. Delete the image by specifying the digest of the image to delete.

```
$ aws ecr batch-delete-image --repository-name my-repo --image-ids
imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304c7c2c1a9d6
fa3e9de6bf552d
```

6.3 Retagging an Image

1. Use the batch-get-image command to get the image manifest for the image to retag and write it to an environment variable.

```
MANIFEST=$(aws ecr batch-get-image --repository-name amazonlinux --
image-ids imageTag=latest --query 'images[].imageManifest' --output
text)
```

2. Use the `--image-tag` option of the **put-image** command to put the image manifest to Amazon ECR with a new tag.

```
aws ecr put-image --repository-name amazonlinux --image-tag 2017.03 -
-image-manifest "$MANIFEST"
```

3. Verify that your new image tag is attached to your image.

```
aws ecr describe-images --repository-name amazonlinux
```

Output:

```
{
    "imageDetails": [
        {
            "imageSizeInBytes": 98755613,
            "imageDigest":
"sha256:8d00af8f076eb15a33019c2a3e7f1f655375681c4e5be157a2685dfe6f247
227",
            "imageTags": [
                "latest",
                "2017.03"
            ],
            "registryId": "aws_account_id",
            "repositoryName": "amazonlinux",
            "imagePushedAt": 1499287667.0
        }
    ]
}
```

6.4 Image Tag Mutability

You can configure a repository to be immutable to prevent image tags from being overwritten. Once the repository is configured for immutable tags, an `ImageTagAlreadyExistsException` error will be returned if you attempt to push an image with a tag that already exists in the repository.

1. To create a repository with immutable tags configured

```
$ aws ecr create-repository --repository-name name --image-tag-
mutability IMMUTABLE --region us-east-2
```

2. To update the image tag mutability settings for an existing repository

```
$ aws ecr put-image-tag-mutability --repository-name name --image-tag-
mutability IMMUTABLE --region us-east-2
```

6.5 Image Scanning

Amazon ECR image scanning helps in identifying software vulnerabilities in your container images. Amazon ECR uses the Common Vulnerabilities and Exposures (CVEs) database from the open source CoreOS Clair project and provides you with a list of scan findings. You can review the scan findings for information about the security of the container images that are being deployed.

1. When a new repository is configured to scan on push, all new images pushed to the repository will be scanned.

   ```
   $ aws ecr create-repository --repository-name name --image-scanning-configuration scanOnPush=true --region us-east-2
   ```

2. Your existing repositories can be configured to scan images when you push them to a repository. This setting will apply to future image pushes.

   ```
   $ aws ecr put-image-scanning-configuration --repository-name name --image-scanning-configuration scanOnPush=true --region us-east-2
   ```

3. You can start image scans manually when you want to scan images in repositories that are not configured to **scan on push**. An image can only be scanned once per day.

   ```
   $ aws ecr start-image-scan --repository-name name --image-id imageTag=tag_name --region us-east-2

   (or)

   $ aws ecr start-image-scan --repository-name name --image-id imageDigest=sha256_hash --region us-east-2
   ```

4. **Retrieving Scan Findings**: You can retrieve the scan findings for the last completed image scan. The findings list by severity the software vulnerabilities that were discovered, based on the Common Vulnerabilities and Exposures (CVEs) database.

   ```
   $ aws ecr describe-image-scan-findings --repository-name name --image-id imageTag=tag_name --region us-east-2
   ```

7. Using the AWS CLI with Amazon ECR

Step 1: Authenticate Docker to your Default Registry

1. Run the aws ecr get-login command.

   ```
   $aws ecr get-login --region region --no-include-email
   ```

   Output:

   ```
   docker login -u AWS -p password https://aws_account_id.dkr.ecr.us-east-1.amazonaws.com
   ```

2. Copy and paste the docker login command into a terminal to authenticate your Docker CLI to the registry. This command provides an authorization token that is valid for the specified registry for 12 hours.

Step 2: Get a Docker Image

- Before you can push an image to Amazon ECR, you must have one to push. If you do not already have an image to use, you can create one by following the steps.

```
$ docker pull ubuntu:trusty
trusty: Pulling from library/ubuntu
0a85502c06c9: Pull complete
0998bf8fb9e9: Pull complete
a6785352b25c: Pull complete
e9ae3c220b23: Pull complete
Digest:
sha256:3cb273da02362a6e667b54f6cf907edd5255c706f9de279c97cfccc7c69881
24
Status: Downloaded newer image for ubuntu:trusty
```

Step 3: Create a Repository

- Now that you have an image to push to Amazon ECR, you must create a repository to hold it.

```
$ aws ecr create-repository --repository-name ubuntu
{
    "repository": {
        "registryId": "111122223333",
        "repositoryName": "ubuntu",
        "repositoryArn":                        "arn:aws:ecr:us-east-
1:111122223333:repository/ubuntu"
    }
}
```

Step 4: Push an Image to Amazon ECR

- Now you can push your image to the Amazon ECR repository you created in the previous section.
    - The minimum version of **docker** is installed: 1.7
    - The Amazon ECR authorization token has been configured with **docker login**.
    - The Amazon ECR repository exists and the user has access to push to the repository.
- List the images you have stored locally to identify the image to tag and push.

```
$ docker images

REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
ubuntu              trusty              e9ae3c220b23        3 weeks ago         187.9 MB
```

- Tag the image to push to your repository.

```
$    docker    tag    ubuntu:trusty    aws_account_id.dkr.ecr.us-east-
1.amazonaws.com/ubuntu:trusty
```

- Push the image

```
$ docker push aws_account_id.dkr.ecr.us-east-
1.amazonaws.com/ubuntu:trusty

The  push  refers  to  a  repository  [aws_account_id.dkr.ecr.us-east-
1.amazonaws.com/ubuntu] (len: 1)
e9ae3c220b23: Pushed
a6785352b25c: Pushed
0998bf8fb9e9: Pushed
0a85502c06c9: Pushed
trusty:                                                        digest:
sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c8
9b size: 6774
```

Step 5: Pull an Image from Amazon ECR

- After your image has been pushed to your Amazon ECR repository, you can pull it from other locations.

```
$ docker pull aws_account_id.dkr.ecr.us-east-
1.amazonaws.com/ubuntu:trusty

trusty: Pulling from ubuntu
0a85502c06c9: Pull complete
0998bf8fb9e9: Pull complete
a6785352b25c: Pull complete
e9ae3c220b23: Pull complete
Digest:
sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c8
9b
Status:  Downloaded  newer  image  for  aws_account_id.dkr.ecr.us-east-
1.amazonaws.com/ubuntu:trusty
```

Step 6: Delete an Image

- If you decide that you no longer need or want an image in one of your repositories, you can delete it with the `batch-delete-image` command.

```
$  aws  ecr  batch-delete-image  --repository-name  ubuntu  --image-ids
imageTag=trusty

{
    "failures": [],
    "imageIds": [
        {
            "imageTag": "trusty",
            "imageDigest":
"sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c
89b"
        }
    ]
}
```

Step 7: Delete a Repository

- ➢ If you decide that you no longer need or want an entire repository of images, you can delete the repository.
- ➢ By default, you cannot delete a repository that contains images; however, the --force flag allows this.

```
$ aws ecr delete-repository --repository-name ubuntu –force

{
    "repository": {
        "registryId": "aws_account_id",
        "repositoryName": "ubuntu",
        "repositoryArn": "arn:aws:ecr:us-east-
1:aws_account_id:repository/ubuntu",
        "createdAt": 1457671643.0,
        "repositoryUri": "aws_account_id.dkr.ecr.us-east-
1.amazonaws.com/ubuntu"
    }
}
```

8. How to push Docker images to AWS ECR

- ➢ To create a repository, it's as simple as executing the following `aws ecr` command:

```
$ aws ecr create-repository --repository-name randserver
```

- ➢ This will create a repository for storing our `randserver` application. Its output should look like this:

```
{
 "repository": {
        "repositoryArn": "arn:aws:ecr:eu-central-
1:123456789012:repository/randserver",
        "registryId": "123456789012",
        "repositoryName": "randserver",
        "repositoryUri": "123456789012.dkr.ecr.eu-central-
1.amazonaws.com/randserver",
     "createdAt": 1543162198.0
 }
}
```

- ➢ A nice addition to your repositories is a life cycle policy that cleans up older versions of your images so that you don't eventually get blocked from pushing a newer version.

```
$ aws ecr put-lifecycle-policy --registry-id 123456789012 --
repository-name randserver --lifecycle-policy-text
'{"rules":[{"rulePriority":10,"description":"Expire old
images","selection":{"tagStatus":"any","countType":"imageCountMoreTha
n","countNumber":800},"action":{"type":"expire"}}]}'
```

- This particular policy will start cleaning up once have more than 800 images on the same repository. You could also clean up based on the images, age, or both, as well as consider only some tags in your cleanup.

- In order use your newly-created ECR repository, first we're going to need to authenticate your local Docker daemon against the ECR registry.

```
$aws ecr get-login --registry-ids 123456789012 --no-include-email
```

- This will output a `docker login` command that will add a new user-password pair for your Docker configuration. You can copy-paste that command, or you can just run it as follows; the results will be the same:

```
$(aws ecr get-login --registry-ids 123456789012 --no-include-email)
```

- Now, pushing and pulling images is just like using any other Docker registry, using the outputted repository URI that we got when creating the repository:

```
$ docker push 123456789012.dkr.ecr.eu-central-
1.amazonaws.com/randserver:0.0.1

$ docker pull 123456789012.dkr.ecr.eu-central-
1.amazonaws.com/randserver:0.0.1
```

- Policies can be attached to an IAM user as follows

```
$ aws iam attach-user-policy --policy-arn
arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly  --user-
name amar
```

- we will create an IAM group for the developers of our `randserver` application:

```
$ aws iam create-group --group-name randserver-developers
    {
          "Group": {
          "Path": "/",
          "GroupName": "randserver-developers",
          "GroupId": "AGPAJRDMVLGOJF3ARET5K",
          "Arn": "arn:aws:iam::123456789012:group/randserver-
developers",
          "CreateDate": "2018-10-25T11:45:42Z"
          }
    }
```

- Then we'll add the `amar` user to the group:

```
$ aws iam add-user-to-group --group-name randserver-developers --user-
name amar
```

- Now we'll need to create our policy so that we can attach it to the group. Copy this JSON document to a file:

```
{
   "Version": "2012-10-17",
   "Statement": [{
         "Effect": "Allow",
         "Action": [
                  "ecr:GetAuthorizationToken",
                  "ecr:BatchCheckLayerAvailability",
                  "ecr:GetDownloadUrlForLayer",
                  "ecr:GetRepositoryPolicy",
                  "ecr:DescribeRepositories",
                  "ecr:ListImages",
                  "ecr:DescribeImages",
                  "ecr:BatchGetImage",
                  "ecr:InitiateLayerUpload",
                  "ecr:UploadLayerPart",
                  "ecr:CompleteLayerUpload",
                  "ecr:PutImage"
             ],
            "Resource": "arn:aws:ecr:eu-central-
1:123456789012:repository/randserver"
   }]
}
```

- To create the policy, execute the following, passing the appropriate path for the JSON document file:

```
$ aws iam create-policy --policy-name EcrPushPullRandserverDevelopers
--policy-document file://./policy.json

    {
          "Policy": {
          "PolicyName": "EcrPushPullRandserverDevelopers",
          "PolicyId": "ANPAITNBFTFWZMI4WFOY6",
          "Arn":
"arn:aws:iam::123456789012:policy/EcrPushPullRandserverDevelopers",
          "Path": "/",
          "DefaultVersionId": "v1",
          "AttachmentCount": 0,
          "PermissionsBoundaryUsageCount": 0,
          "IsAttachable": true,
          "CreateDate": "2018-10-25T12:00:15Z",
          "UpdateDate": "2018-10-25T12:00:15Z"
          }
    }
```

- The final step is then to attach the policy to the group, so that `amar` and all future developers of this application can use the repository from their workstation:

```
$ aws iam attach-group-policy --group-name randserver-developers --
policy-arn
arn:aws:iam::123456789012:policy/EcrPushPullRandserverDevelopers
```

9. Reference

- https://docs.aws.amazon.com/AmazonECR

- Deploy Docker Containers on Amazon Elastic Container Service (Amazon ECS): https://aws.amazon.com/getting-started/tutorials/deploy-docker-containers/

- How to push Docker images to AWS' Elastic Container Registry(ECR): https://hub.packtpub.com/how-to-push-docker-images-to-aws-elastic-container-registryecr-tutorial/