| **Docker with Amazon ECR and Amazon Linux** **1. Introduction** Docker is a platform that enables developers to package applications into containers—standardized units that include everything the software needs to run, including libraries, system tools, code, and runtime. Amazon Elastic Container Registry (ECR) is a fully managed Docker container registry that makes it easy to store, manage, and deploy Docker container images. Amazon Linux is a Linux distribution optimized for AWS that provides a secure, stable, and high-performance environment for running applications in the cloud. **2. Objectives** The goal of this documentation is to guide you through the process of:   1. Setting up an Amazon Linux instance on AWS. 2. Installing Docker on Amazon Linux. 3. Creating a Docker image and pushing it to Amazon ECR. 4. Pulling and running the Docker image from ECR on an Amazon Linux instance.  **3. Architecture** The architecture of this setup involves the following key components:   * **Amazon EC2 (Elastic Compute Cloud):** Running Amazon Linux to host and execute Docker containers. * **Amazon ECR (Elastic Container Registry):** For storing Docker images. * **Docker:** For containerizing applications and running them on Amazon Linux.   lua  Copy code  +------------------------------------+  | AWS Environment |  | |  | +----------------------------+ |  | | Amazon ECR | |  | | (Docker Image Storage) | |  | +----------------------------+ |  | |  | +----------------------------+ |  | | Amazon EC2 Instance | |  | | (Amazon Linux + Docker) | |  | +----------------------------+ |  | |  +------------------------------------+ **4. Steps****Step 1: Launch an Amazon Linux Instance**  1. Log in to your AWS Management Console. 2. Navigate to the **EC2 Dashboard** and click **Launch Instance**. 3. Choose **Amazon Linux 2 AMI** as the Amazon Machine Image (AMI). 4. Select an instance type (e.g., t2.micro for free tier). 5. Configure the instance (VPC, subnet, IAM role, etc.). 6. Add storage if needed. 7. Configure the security group to allow SSH access. 8. Review and launch the instance, then connect via SSH.  **Step 2: Install Docker on Amazon Linux** Update the package manager: bash Copy code sudo yum update -y  Install Docker: bash Copy code sudo amazon-linux-extras install docker  Start the Docker service: bash Copy code sudo service docker start  Add your user to the Docker group to run Docker commands without sudo: bash Copy code sudo usermod -aG docker $USER **Step 3: Create and Push a Docker Image to Amazon ECR** Authenticate Docker to the Amazon ECR registry: bash Copy code aws ecr get-login-password --region <your-region> | docker login --username AWS --password-stdin <aws\_account\_id>.dkr.ecr.<your-region>.amazonaws.com  Create a Dockerfile for your application: dockerfile Copy code # Sample Dockerfile  FROM amazonlinux:2  RUN yum -y install httpd  CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]  Build the Docker image: bash Copy code docker build -t my-app .  Tag the Docker image for ECR: bash Copy code docker tag my-app:latest <aws\_account\_id>.dkr.ecr.<your-region>.amazonaws.com/my-app:latest  Push the Docker image to Amazon ECR: bash Copy code docker push <aws\_account\_id>.dkr.ecr.<your-region>.amazonaws.com/my-app:latest **Step 4: Pull and Run the Docker Image on Amazon Linux** Pull the Docker image from ECR: bash Copy code docker pull <aws\_account\_id>.dkr.ecr.<your-region>.amazonaws.com/my-app:latest  Run the Docker container: bash Copy code docker run -d -p 80:80 <aws\_account\_id>.dkr.ecr.<your-region>.amazonaws.com/my-app:latest   1. Verify the application is running by accessing the EC2 instance's public IP in your browser.  **5. Conclusion** By following this guide, you have successfully set up a containerized application on an Amazon Linux instance, stored the Docker image in Amazon ECR, and deployed the application from ECR. This process illustrates the seamless integration of Docker with AWS services, enabling scalable and efficient application deployment in the cloud.              Creating Ec2 Instance:-                Commands:-          vi dockerfile      Creating an ECR:          Login with ECR command:-        Change the image Version:-            To delete the image from ec2:-      1 yum update -y  2 yum upgrade -y  3 yum install docker  4 service docker status  5 service docker start  6 service docker enable  7 service docker status  8 touch dockerfile  9 cat dockerfile  10 ls  11 vi dockerfile  12 cat dockerfile  13 docker images  14 docker build -t amit\_image .  15 docker images  16 aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 891377318947.dkr.ecr.us-east-2.amazonaws.com  17 docker images  18 docker tag amit\_image:latest 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:latest  19 docker images  20 docker push 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo  21 vi dockerfile  22 cat dockerfile  23 docker images  24 docker build -t image1:version1  25 docker build -t image1:version1 .  26 docker images  27 docker build -t amit\_image:version1 .  28 docker images  29 aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 891377318947.dkr.ecr.us-east-2.amazonaws.com  30 docker tag amit\_image:version1 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:version1  31 docker images  32 docker push 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo  33 docker push 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:version1  34 docker rmi amit\_image:latest  35 docker images  36 docker images  37 docker rmi amit\_image:version1  38 docker rmi 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:version1  39 docker images  40 docker rmi 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:latest  41 docker rmi image1:latest  42 docker images  43 docker rmi image1:latest  44 docker images  45 docker pull 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:latest  46 docker images  47 docker pull 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:version1  48 docker images  49 docker run -d -p 80:80 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:latest  50 docker ps  51 docker run -d -p 80:80 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:version1  52 docker ps  53 docker container inspect cff23b1becde  54 curl http://172.17.0.2  55 history  56 curl http://172.17.0.2  57 docker images  58 docker container run -it 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:latest /bin/bash  59 docker container run -it 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:latest /bin/bash  60 sudo amazon-linux-extras install docker  61 amazon-linux-extras install docker  62 history  63 docker container run -it 891377318947.dkr.ecr.us-east-2.amazonaws.com/amit\_naidu\_repo:latest /bin/bash  64 history **Docker with ECR & ECS Documentation****Objective** The goal of this task is to deploy a Docker containerized application on an AWS EC2 instance, push the Docker image to Amazon Elastic Container Registry (ECR), and manage the container using Amazon Elastic Container Service (ECS). The task involves setting up the EC2 instance, configuring Docker, creating and pushing Docker images to ECR, and using ECS for container orchestration. **Architecture Flow**  1. **EC2 Instance Setup**:    * Launch an EC2 instance using Amazon Linux 2.    * Install Docker on the EC2 instance.    * Use the Dockerfile to build a Docker image on the EC2 instance. 2. **ECR (Elastic Container Registry) Configuration**:    * Create a private repository in ECR to store Docker images.    * Set up IAM roles with EC2InstanceProfile, ECRFullAccess, and ECRPublicFullAccess to allow the EC2 instance to push and pull images from ECR. 3. **Pushing Docker Images to ECR**:    * Tag and push the built Docker image from the EC2 instance to the ECR repository. 4. **ECS (Elastic Container Service) Integration**:    * Use ECS to manage and deploy Docker containers using the images stored in ECR.  **Steps**  1. **EC2 Instance Setup**:   Update the EC2 instance: bash Copy code yum update -y  yum upgrade  Install Docker: bash Copy code yum install docker -y  service docker start  service docker enable  Create and edit the Dockerfile: bash Copy code touch Dockerfile  vi Dockerfile  Build the Docker image: bash Copy code docker build -t image1 .  List Docker images: bash Copy code docker images   1. **ECR Configuration**:    * Create a private ECR repository.    * Enable AES encryption for the repository.    * Assign the necessary IAM roles (ECRFullAccess and ECRPublicFullAccess) to the EC2 instance. 2. **Pushing Docker Images to ECR**:   Tag the Docker image: bash Copy code docker tag image1:latest <repository\_uri>:tag  Push the image to ECR: bash Copy code docker push <repository\_uri>:tag   1. **ECS Configuration**:    * Use the images in ECR to manage Docker containers in ECS.  **Conclusion** This setup allows you to efficiently manage and deploy Docker containers using AWS services like EC2, ECR, and ECS. By storing your Docker images in a private ECR repository, you ensure security and ease of access for deployment. ECS provides scalable and flexible management of your containerized applications. **Use Case Scenario** This setup is particularly useful for teams looking to implement continuous deployment of containerized applications on AWS. By integrating Docker with AWS services, you can automate the deployment pipeline, scale applications as needed, and maintain secure and reliable infrastructure for your applications. |
| --- |