**1. Aamir khan**

**Deploy a Static Website on AWS S3 and Use CloudFront**

* **Procedure**:
  1. Create an S3 bucket and enable static website hosting.
  2. Upload your static website files (HTML, CSS, JS) to the S3 bucket.
  3. Configure CloudFront to use the S3 bucket as the origin.
  4. Test the website with the CloudFront URL.

**2. Abhinav Kumar Singh**

**Dockerize a Node.js Application and Deploy on EC2**

* **Procedure**:
  1. Create a simple Node.js application.
  2. Write a Dockerfile to containerize the application.
  3. Build and push the Docker image to Docker Hub.
  4. Launch an EC2 instance and install Docker.
  5. Pull and run the Docker container on EC2.

**3. Abhishek Kumar**

**CI/CD Pipeline with Jenkins and Docker on AWS**

* **Procedure**:
  1. Set up a Jenkins server on an EC2 instance.
  2. Create a Dockerfile for a sample application.
  3. Configure Jenkins to build Docker images and push them to Docker Hub.
  4. Deploy the Docker container to another EC2 instance.
  5. Automate the deployment process using Jenkins pipelines.

**4. Abhishek kumar T22EJICS010**

**Build a Python Flask App and Deploy Using AWS Elastic Beanstalk**

* **Procedure**:
  1. Develop a simple Python Flask application.
  2. Install the AWS Elastic Beanstalk CLI and initialize the environment.
  3. Deploy the Flask application using Elastic Beanstalk.
  4. Monitor and scale the application using the AWS Management Console.

**5. Abhishek Tanwar**

**Docker Swarm Cluster on AWS**

* **Procedure**:
  1. Launch multiple EC2 instances to serve as Swarm manager and worker nodes.
  2. Install Docker on all instances.
  3. Initialize Docker Swarm on the manager node.
  4. Join worker nodes to the Swarm.
  5. Deploy a multi-container application using Docker Swarm.

**6. Aditya Bhardwaj**

**Set Up a Virtual Private Network (VPN) on a Docker Container**

* **Procedure**:
  1. Create a Dockerfile to set up a VPN server.
  2. Build and run the Docker container with the VPN configuration.
  3. Configure the host machine to connect through the VPN container.
  4. Verify the VPN connection and test the network traffic.

**7. Akshay Dusane**

**Kubernetes on AWS with EKS**

* **Procedure**:
  1. Create an Amazon EKS cluster using the AWS Management Console.
  2. Configure kubectl to interact with the EKS cluster.
  3. Deploy a sample application using Kubernetes manifests.
  4. Scale the application and manage resources within the cluster.
  5. Set up monitoring and logging with AWS CloudWatch.

**8. Ambika Mathur**

**Implement Continuous Deployment with GitLab CI/CD and Docker**

* **Procedure**:
  1. Set up a GitLab repository for the project.
  2. Write a Dockerfile for the application.
  3. Configure GitLab CI/CD pipelines to build, test, and deploy the Docker container.
  4. Deploy the container to a Docker host (e.g., EC2 instance).
  5. Set up automated tests and deployments upon code changes.

9. Anchal Pareek

**Serverless Image Processing with AWS Lambda and S3**

* **Procedure**:
  1. Create an S3 bucket to store uploaded images.
  2. Write an AWS Lambda function to process images (e.g., resizing).
  3. Set up an S3 trigger to invoke the Lambda function upon image upload.
  4. Deploy the Lambda function and test the image processing workflow.
  5. Store the processed images in another S3 bucket.

**10. Anuradha bhagat**

**Multi-Cloud Deployment with Docker and Terraform**

* **Procedure**:
  1. Define infrastructure using Terraform for multiple cloud providers (e.g., AWS and Azure).
  2. Create Docker images for the application.
  3. Use Terraform to deploy the application containers across the cloud providers.
  4. Implement load balancing and monitoring across the deployment.
  5. Test failover and scalability between the cloud environments.

**11. Arnav Mishra**

**Dockerized Microservices Architecture on AWS ECS**

* **Procedure**:
  1. Develop multiple microservices, each with its own Dockerfile.
  2. Push Docker images to Amazon ECR (Elastic Container Registry).
  3. Set up an ECS cluster and define tasks for each microservice.
  4. Deploy the services and configure service discovery.
  5. Implement inter-service communication and monitor performance.

**12. Ashish Pandey**

**Automate Infrastructure with AWS CloudFormation and Docker**

* **Procedure**:
  1. Write a CloudFormation template to provision AWS resources.
  2. Include EC2 instances, S3 buckets, and security groups in the template.
  3. Automate the deployment of Docker containers on the provisioned EC2 instances.
  4. Test and validate the infrastructure setup.
  5. Use CloudFormation stack updates for infrastructure changes.

**13. Ashmit kumar**

**Deploy a WordPress Site Using Docker and AWS RDS**

* **Procedure**:
  1. Create a Docker Compose file for WordPress and MySQL containers.
  2. Set up an RDS instance for the WordPress database.
  3. Configure WordPress to connect to the RDS instance.
  4. Deploy the Docker containers on an EC2 instance.
  5. Set up backups and security measures for the RDS database**.**

**14. Bhagyashree Mandlawat**

**Serverless REST API with API Gateway and AWS Lambda**

* **Procedure**:
  1. Write Lambda functions for API endpoints (CRUD operations).
  2. Set up an API Gateway to route requests to Lambda functions.
  3. Deploy the API and test the endpoints.
  4. Implement authentication using AWS Cognito.
  5. Monitor API usage and performance.

**15. Bharat Sain**

**Build a Chatbot with AWS Lex and Lambda**

* **Procedure**:
  1. Design a conversational flow for the chatbot.
  2. Create a Lex bot and define intents, slots, and responses.
  3. Write Lambda functions to handle backend logic.
  4. Integrate the Lex bot with a messaging platform (e.g., Slack).
  5. Test and refine the chatbot's responses.

**16. Bittu Kumar**

**Monitoring Docker Containers with Prometheus and Grafana**

* **Procedure**:
  1. Set up Prometheus to collect metrics from Docker containers.
  2. Configure Grafana to visualize metrics from Prometheus.
  3. Deploy Prometheus and Grafana using Docker Compose.
  4. Create dashboards and set up alerts for container metrics.
  5. Test the monitoring setup and analyze the metrics.

**17. Divyanshi Arora**

**Automate Docker Image Build and Deployment with GitHub Actions**

* **Procedure**:
  1. Create a GitHub repository and add a Dockerfile.
  2. Write a GitHub Actions workflow to build and push Docker images.
  3. Configure the workflow to deploy the Docker container to a server.
  4. Implement automated tests as part of the CI/CD pipeline.
  5. Monitor the deployment process and handle errors.

**18. Divyanshu Yadav**

**AWS Fargate for Serverless Container Hosting**

* **Procedure**:
  1. Create a Docker image for the application.
  2. Push the image to Amazon ECR.
  3. Set up an AWS Fargate task definition with the Docker image.
  4. Deploy the task in an ECS cluster.
  5. Monitor and scale the application based on demand.

**19. Garvit Jain**

**Secure Docker Containers with AWS Secrets Manager**

* **Procedure**:
  1. Create secrets (e.g., API keys, database passwords) in AWS Secrets Manager.
  2. Modify the Docker application to retrieve secrets from Secrets Manager.
  3. Deploy the Docker container on an EC2 instance.
  4. Test the application to ensure secure access to secrets.
  5. Implement rotation and management of secrets.

**20. Gunjan Pareek**

**Data Pipeline with AWS S3, Lambda, and DynamoDB**

* **Procedure**:
  1. Create an S3 bucket to store data files.
  2. Set up a Lambda function to process data and store it in DynamoDB.
  3. Configure an S3 trigger to invoke the Lambda function on file upload.
  4. Create a DynamoDB table for processed data.
  5. Monitor the data flow and verify data accuracy.

**21. Harish kumar**

**Multi-Tier Architecture with Docker and AWS RDS**

* **Procedure**:
  1. Create a Docker container for the web server (e.g., Nginx).
  2. Set up an RDS instance for the database layer.
  3. Configure the Docker container to connect to the RDS database.
  4. Deploy the application on an EC2 instance.
  5. Implement security groups and network ACLs for the architecture.

**22. Harshit jain**

**AWS Cloud9 IDE for Containerized Development**

* **Procedure**:
  1. Set up an AWS Cloud9 environment.
  2. Clone a Git repository with Docker configurations.
  3. Develop and test the application within the Cloud9 IDE.
  4. Build and run Docker containers from Cloud9.
  5. Deploy the application from Cloud9 to an EC2 instance.

**23. Hitendra Singh**

**Implementing CI/CD with AWS CodePipeline and Docker**

* **Procedure**:
  1. Set up a CodeCommit repository for source code.
  2. Create a CodePipeline to build Docker images and deploy them.
  3. Integrate CodeBuild to build the Docker image.
  4. Use CodeDeploy to deploy the Docker container to EC2.
  5. Monitor the CI/CD process and handle deployment failures.

**24. Jahnavi Joshi**

**Serverless Image Recognition with AWS Rekognition and Lambda**

* **Procedure**:
  1. Create an S3 bucket for image uploads.
  2. Write a Lambda function to process images using AWS Rekognition.
  3. Set up an S3 trigger to invoke the Lambda function.
  4. Deploy the Lambda function and test image recognition.
  5. Store and display recognition results.

**25. Jiya Arora**

**Implementing Infrastructure as Code with Terraform and Docker**

* **Procedure**:
  1. Write Terraform scripts to provision AWS resources.
  2. Include Docker containers in the Terraform configuration.
  3. Use Terraform to deploy the infrastructure and Docker containers.
  4. Monitor and manage the deployed infrastructure.
  5. Implement version control for the Terraform scripts.

**26. Jyotish kumar**

**Automate Virtual Machine Deployment with Vagrant and Docker**

* **Procedure**:
  1. Create a Vagrantfile to define a virtual machine environment.
  2. Include Docker installation and configuration in the Vagrantfile.
  3. Automate the setup of a Docker container within the VM.
  4. Test the deployment and functionality of the Docker container.
  5. Modify and extend the Vagrantfile for additional configurations.

**27. Kamlesh Kumar**

**Hybrid Cloud Application with AWS and On-Premises Virtual Machines**

* **Procedure**:
  1. Set up an on-premises virtual machine running a web server.
  2. Deploy a database on AWS RDS.
  3. Configure the web server to connect to the RDS instance.
  4. Implement VPN connectivity between the on-premises network and AWS.
  5. Monitor and secure the hybrid cloud setup.

**28. Kaushal Kumar**

**Build a Machine Learning Model with AWS SageMaker and Deploy with Docker**

* **Procedure**:
  1. Develop and train a machine learning model using AWS SageMaker.
  2. Export the trained model and create a Docker container for serving it.
  3. Push the Docker image to Amazon ECR.
  4. Deploy the model container on an EC2 instance or AWS ECS.
  5. Create an API endpoint for predictions and test the deployment.

**29. kavita bhadral**

**Implement Auto Scaling with Docker and AWS EC2 Auto Scaling**

* **Procedure**:
  1. Create a Docker image for the application.
  2. Set up an EC2 Auto Scaling group with a launch template for the Docker container.
  3. Configure auto-scaling policies based on metrics (e.g., CPU utilization).
  4. Deploy the Docker container and test auto-scaling.
  5. Monitor and adjust scaling policies as needed.

**30. Keshav Shrimali**

**AWS Lambda for Serverless Microservices Architectusre**

* **Procedure**:
  1. Write Lambda functions for different microservices (e.g., user management, payment processing).
  2. Set up API Gateway to route requests to the appropriate Lambda functions.
  3. Deploy the microservices and test the end-to-end workflow.
  4. Implement monitoring and logging with AWS CloudWatch.
  5. Optimize the Lambda functions for performance and cost.

**31. Khushi Meenia**

**Serverless File Processing Pipeline with AWS Step Functions**

* **Procedure**:
  1. Define a workflow in AWS Step Functions for file processing.
  2. Create Lambda functions for each step of the workflow.
  3. Set up an S3 bucket for file uploads and trigger the workflow.
  4. Deploy the Step Functions workflow and Lambda functions.
  5. Monitor and troubleshoot the workflow execution.

**32. kishan raj**

**Deploy a Dockerized Django Application on AWS Lightsail**

* **Procedure**:
  1. Develop a Django application and create a Dockerfile.
  2. Push the Docker image to a container registry.
  3. Create a Lightsail instance and install Docker.
  4. Deploy the Docker container on the Lightsail instance.
  5. Set up domain and SSL for the Django application.

**33. Krishna Kumar**

**Implement Blue/Green Deployments with Docker and AWS**

* **Procedure**:
  1. Set up two identical environments (blue and green) using Docker containers.
  2. Deploy the current version of the application to the blue environment.
  3. Deploy a new version to the green environment.
  4. Switch traffic from blue to green using a load balancer.
  5. Monitor the deployment and roll back if necessary.

**34. krishna Patel**

**AWS Batch for Batch Processing of Docker Containers**

* **Procedure**:
  1. Create a Docker image for the batch processing job.
  2. Set up an AWS Batch compute environment and job queue.
  3. Define a job definition with the Docker image.
  4. Submit batch jobs to the queue and monitor their execution.
  5. Handle job retries and failures.

**35. Megha gupta**

**Secure AWS Infrastructure with Docker and IAM Roles**

* **Procedure**:
  1. Define IAM roles and policies for different AWS services.
  2. Deploy Docker containers on EC2 instances with specific IAM roles.
  3. Restrict access to AWS resources using IAM policies.
  4. Monitor and audit access to the infrastructure.
  5. Implement best practices for security and compliance.

**36. Meghna Vyas**

**Deploy a Multi-Region Application with Docker and AWS Route 53**

* **Procedure**:
  1. Create Docker images for the application and push them to ECR.
  2. Set up EC2 instances in multiple AWS regions.
  3. Deploy the Docker containers in each region.
  4. Configure Route 53 for DNS failover and latency-based routing.
  5. Test the multi-region setup and handle failover scenarios.

**37. MOHAMMED SAIF**

**AWS CloudTrail for Monitoring Docker Container Activity**

* **Procedure**:
  1. Enable AWS CloudTrail for logging API activity.
  2. Deploy Docker containers on EC2 instances.
  3. Configure CloudTrail to log Docker-related API calls.
  4. Analyze the logs for security and compliance purposes.
  5. Set up alerts for suspicious activities.

**38.** MOKSH SHARMA

**Automate Docker Container Deployment with AWS OpsWorks**

* **Procedure**:
  1. Create an OpsWorks stack and layer for the Docker application.
  2. Define OpsWorks recipes for deploying Docker containers.
  3. Use OpsWorks to manage EC2 instances and deploy the application.
  4. Monitor the deployment and scale the application as needed.
  5. Implement automated backups and updates.

**39. Murli Manohar Soni**

**Implement Infrastructure Monitoring with Docker, Prometheus, and Grafana**

* **Procedure**:
  1. Set up Docker containers for Prometheus and Grafana.
  2. Configure Prometheus to collect metrics from other Docker containers.
  3. Visualize metrics and set up dashboards in Grafana.
  4. Set up alerts for critical infrastructure events.
  5. Monitor and optimize the infrastructure based on metrics.

**40. Naman Sahai**

**Serverless Data Processing with AWS Lambda and Kinesis**

* **Procedure**:
  1. Set up a Kinesis stream for real-time data ingestion.
  2. Write a Lambda function to process data from the stream.
  3. Configure the Lambda function as a Kinesis stream consumer.
  4. Deploy the Lambda function and test the data processing workflow.
  5. Store the processed data in an S3 bucket or database.

**41. Pankaj Kumar**

**Dockerized Web Application with AWS RDS and CloudFront**

* **Procedure**:
  1. Create a Docker container for the web application.
  2. Set up an RDS instance for the database.
  3. Deploy the Docker container on an EC2 instance and connect to RDS.
  4. Configure CloudFront for content delivery and caching.
  5. Monitor application performance and optimize as needed.

**42. Prateek**

**AWS Lambda for Event-Driven Microservices Architecture**

* **Procedure:**
  1. Write Lambda functions for different microservices (e.g., notification service).
  2. Use Amazon SNS for inter-service communication.
  3. Set up S3 triggers and DynamoDB Streams to invoke Lambda functions.
  4. Deploy the microservices and test the event-driven architecture.
  5. Monitor and scale the architecture based on usage.

**43. Prince kumar**

**Secure Web Application with Docker, Nginx, and AWS WAF**

* **Procedure:**
  1. **Develop a web application and create a Dockerfile.**
  2. **Set up an Nginx container as a reverse proxy for the application.**
  3. **Deploy the Docker containers on EC2 instances.**
  4. **Configure AWS WAF to protect the application from common threats.**
  5. **Monitor traffic and adjust WAF rules as needed.**

**44. Priyani vyas**

**Serverless ETL Pipeline with AWS Glue and Lambda**

* **Procedure**:
  1. Define an ETL job in AWS Glue for data transformation.
  2. Create a Lambda function to trigger the Glue job.
  3. Set up a schedule or event trigger for the Lambda function.
  4. Deploy the ETL pipeline and test data processing.
  5. Store the transformed data in a data warehouse or S3 bucket.

**45. Radhika sen**

**Multi-Stage Docker Builds and Deployment on AWS ECS**

* **Procedure**:
  1. Create a multi-stage Dockerfile for the application.
  2. Build and push the Docker image to ECR.
  3. Set up an ECS cluster and task definition.
  4. Deploy the Docker container on ECS and test the application.
  5. Implement monitoring and scaling for the ECS services.

**46. Rahul choudhary**

**AWS Step Functions for Orchestrating Serverless Workflows**

* **Procedure**:
  1. Define a workflow in AWS Step Functions.
  2. Create Lambda functions for each step in the workflow.
  3. Deploy the workflow and test the orchestration.
  4. Monitor the workflow execution and handle errors.
  5. Optimize the workflow for performance and cost.

**47. Rahul Kumar Gond**

**CI/CD with Docker, AWS CodePipeline, and CodeDeploy**

* **Procedure**:
  1. Set up a CodeCommit repository for the application.
  2. Write a Dockerfile and create a Docker image.
  3. Configure CodePipeline to build, test, and deploy the Docker image.
  4. Use CodeDeploy to deploy the application to EC2 instances.
  5. Monitor the CI/CD pipeline and handle deployment failures.

**48. Rajat Sen**

**AWS App Runner for Simple Containerized Application Deployment**

* **Procedure**:
  1. Create a Docker image for the application and push it to ECR.
  2. Set up AWS App Runner with the Docker image.
  3. Deploy the application and test the deployment.
  4. Monitor and scale the application using App Runner.
  5. Implement security and monitoring best practices.

**49. Rajeev kumar**

**Deploy a Dockerized E-commerce Platform on AWS ECS**

* **Procedure**:
  1. Develop or use an open-source e-commerce platform and create Docker images for its components.
  2. Set up an ECS cluster and task definitions for each component (e.g., web server, database).
  3. Deploy the platform using ECS and configure service discovery.
  4. Monitor performance and scale the platform as needed.
  5. Implement security and backup solutions for the platform.

**50. Ravinder Bishnoi**

**Serverless Data Lake with AWS S3, Glue, and Athena**

* **Procedure**:
  1. Create an S3 bucket to store raw data.
  2. Use AWS Glue to catalog and transform the data.
  3. Set up AWS Athena for querying the data lake.
  4. Deploy the data lake infrastructure and ingest data.
  5. Test data queries and analyze the results.

**51. Rishi**

**Build a Scalable Chat Application with Docker and AWS ElastiCache**

* **Procedure**:
  1. Develop a chat application and create Docker images for its components (e.g., frontend, backend).
  2. Set up an ElastiCache cluster for managing session data and real-time communication.
  3. Deploy the application on ECS and connect to ElastiCache.
  4. Test the application's scalability and real-time features.
  5. Implement monitoring and security measures.

**52. Ritik raj singh**

**AWS CloudFormation for Automated Docker Container Deployment**

* **Procedure**:
  1. Write a CloudFormation template to define the infrastructure (e.g., EC2 instances, VPC).
  2. Include Docker container deployment in the CloudFormation stack.
  3. Deploy the stack and test the infrastructure setup.
  4. Use CloudFormation stack updates for infrastructure changes.
  5. Monitor and manage the deployed infrastructure.

**53. Rudra Pratap Singh Bhati**

**Implement Serverless Machine Learning Inference with AWS Lambda**

* **Procedure**:
  1. Train a machine learning model and export it.
  2. Create a Lambda function to load the model and make predictions.
  3. Set up an API Gateway to expose the Lambda function as an API.
  4. Deploy the Lambda function and test the API.
  5. Monitor the API usage and optimize the Lambda function.

**54. Sachin Kumar**

**Build a Real-Time Analytics Dashboard with AWS Kinesis and Lambda**

* **Procedure**:
  1. Set up a Kinesis stream for ingesting real-time data.
  2. Write a Lambda function to process the data and store it in a database.
  3. Create a web dashboard to visualize the real-time analytics.
  4. Deploy the infrastructure and test the real-time data flow.
  5. Monitor and scale the analytics pipeline as needed.

**55. Sarvjeet kumar**

**Implementing a Content Delivery Network (CDN) with AWS CloudFront and S3**

* **Procedure**:
  1. Host static website content in an S3 bucket.
  2. Set up a CloudFront distribution with the S3 bucket as the origin.
  3. Configure caching and security settings for the CloudFront distribution.
  4. Test the CDN by accessing the content via CloudFront.
  5. Monitor and optimize the CDN performance.

**56. Shyam suthar**

**AWS Lambda for Serverless Data Aggregation and Reporting**

* **Procedure**:
  1. Create multiple Lambda functions to aggregate data from various sources.
  2. Use DynamoDB or S3 to store the aggregated data.
  3. Set up a reporting API using API Gateway and Lambda.
  4. Deploy the data aggregation and reporting system.
  5. Monitor the system's performance and optimize as needed.

**57. Sneha Dadhich**

**Automate Docker Container Management with AWS Elastic Beanstalk**

* **Procedure**:
  1. Create a Dockerfile for the application.
  2. Deploy the application using AWS Elastic Beanstalk.
  3. Monitor the environment and scale the application as needed.
  4. Implement rolling updates and versioning.
  5. Set up automated backups and disaster recovery.

**58. Vasudev solanki**

**Build a Serverless IoT Application with AWS IoT and Lambda**

* **Procedure**:
  1. Set up AWS IoT Core to manage IoT devices.
  2. Create a Lambda function to process IoT data.
  3. Set up rules in AWS IoT to trigger the Lambda function.
  4. Deploy the IoT infrastructure and connect devices.
  5. Monitor and analyze the IoT data.

**59. Vinita Deora**

**Secure Docker Containers with AWS IAM and Secrets Manager**

* **Procedure**:
  1. Define IAM roles and policies for accessing AWS services.
  2. Store sensitive information (e.g., API keys) in Secrets Manager.
  3. Modify the Docker application to retrieve secrets securely.
  4. Deploy the application with IAM roles and secure secrets.
  5. Monitor and audit access to secrets and AWS resources.

**60. Yash Kansara**

**Serverless API with AWS Lambda and DynamoDB**

* **Procedure**:
  1. Write Lambda functions to handle API requests (e.g., create, read, update, delete).
  2. Set up DynamoDB tables to store API data.
  3. Use API Gateway to expose the Lambda functions as a REST API.
  4. Deploy the API and test all endpoints.
  5. Implement monitoring and logging for the API.

These projects cover a range of topics and complexities, providing practical experience in using Docker, AWS, and Agile methodologies. Students can start with simpler projects and gradually move on to more complex ones, gaining hands-on experience in real-world scenarios.

Top of Form

Bottom of Form