To deploy dynamic web apps on AWS using CI/CD pipelines and GitHub Actions, follow these steps:

1. **Configure AWS Credentials:**
   * Set up AWS credentials on your local machine using AWS CLI or IAM roles.
2. **Install Terraform on Your Machine:**
   * Download and install Terraform on your local machine from the Terraform website or use a package manager.
3. **Build AWS Infrastructure:**
   * Define your AWS infrastructure using Terraform code (e.g., EC2 instances, RDS database, S3 bucket, ECS Fargate service, etc.).
   * Initialize Terraform and apply the configuration to create the infrastructure on AWS.
4. **Create GitHub Action for Application Deployment:**
   * Create a GitHub repository for your project if you haven't already.
   * Set up GitHub Actions workflow YAML file in your repository to automate deployment.
   * Define steps in the workflow YAML file to trigger deployment based on events such as code pushes or pull requests.
5. **Create ECR Repository:**
   * Use AWS CLI or AWS Management Console to create a repository in Amazon Elastic Container Registry (ECR) to store Docker images.
6. **Start Self-hosted EC2 Server:**
   * If needed, provision an EC2 instance to act as a self-hosted runner for GitHub Actions.
7. **Build and Push Docker Image to ECR:**
   * Set up Dockerfile for your application.
   * Build Docker image locally or as part of GitHub Actions workflow.
   * Push the built Docker image to the ECR repository.
8. **Create Environment File and Export to S3:**
   * Create environment configuration files (e.g., .env) for your application.
   * Upload these files to an S3 bucket to keep them centralized and accessible.
9. **Migrate Data into RDS Database with Flyway:**
   * Set up Flyway for database schema migrations.
   * Create migration scripts to manage database changes.
   * Execute Flyway migrations as part of your deployment process.
10. **Stop Self-hosted EC2 Runner:**
    * Stop or terminate the self-hosted EC2 runner if it's no longer needed to reduce costs.
11. **Create New Task Definition Revision:**
    * Define a new task definition revision in Amazon ECS with updated Docker image details or configurations.
12. **Restart ECS Fargate Service:**
    * Update the ECS service to use the new task definition revision, triggering a restart of your Fargate service with the latest changes.

By following these steps, you can set up a robust CI/CD pipeline using GitHub Actions and AWS services for deploying dynamic web applications. Make sure to test your pipeline thoroughly to ensure smooth deployments.