**GitHub Actions Pipeline Documentation for Drivemate**

**Deployment**

### **Overview**

This documentation provides an in-depth understanding of the Drivemate CI/CD Pipeline using GitHub Actions. It explains why the pipeline was developed, the steps involved, pre-requisites, the overall approach, AWS configuration, secrets management, and instructions for a new user to get started.

### **Why We Developed This Pipeline:**

1. **Automation**:
   * Automate the deployment of different branches to a single EC2 instance with minimal manual intervention.
2. **Dynamic Resource Management**:
   * Simplify and standardize deployment for branch-specific resources such as ports, subdomains, Route53 records, and Docker containers.
3. **Scalability**:
   * Allow multiple branches to run concurrently using unique subdomains and ports.
4. **Visibility**:
   * Enable monitoring of active deployments via config.test.drivemate.au.

### **Pipeline Steps:**

1. **Pipeline Trigger**
   * Triggered when a push is made to the branch.
2. **Build and Push Docker Image**
   * Check out the code from the repository.
   * Configure AWS credentials using OpenID Connect (OIDC).
   * Parse the branch-specific configuration from branches-config.yml.
   * Fetch secrets from AWS Secrets Manager to build the environment.
   * Build and push the Docker image to Amazon ECR.
3. **Deployment to EC2**
   * Fetch deployment secrets from AWS Secrets Manager (e.g., EC2 host, security group, Route53 zone ID).
   * Establish an SSH connection to the EC2 instance using the fetched private key.
   * Dynamically allocate an available port for the branch.
   * Run the Docker container on the EC2 instance with the allocated port.
   * Update Nginx configurations and AWS Route 53 DNS records.
   * Automatically configure subdomains and Route53 records for HTTPS with SSL.
4. **Cleanup Resources**
   * Identify branches no longer present in branches-config.yml.
   * Stop and remove Docker containers and images for unused branches.
   * Delete old Route53 subdomain records.
   * Remove unused ports, Nginx configurations, and listener rules in the load balancer.

### **Prerequisites**

1. **GitHub Repository**:
   * Ensure the repository is set up with necessary branch configurations.
2. **AWS Resources**:
   * EC2 instance with Docker, Nginx, and necessary permissions.
   * Amazon ECR for Docker image storage.
   * AWS Secrets Manager for managing deployment secrets.
   * Route53 for managing DNS records.
   * Load Balancer for HTTPS and domain routing.
3. **AWS Secrets Manager**:
   * Store sensitive information such as private keys, EC2 details, and security group IDs.
4. **GitHub Secrets**:
   * Add required AWS account details and region.
5. **Branch Configuration File** **(**branches-config.yml**)**
   * Define branch-specific deployment configurations like environment, ECR repository, and secrets IDs.

### **Approach**

1. **Dynamic Subdomain Setup:**
   * **Branch Name Simplification**: The branch name is sanitized to remove special characters for use in subdomains.
   * **Port Allocation**: Randomly assigns a port to each branch and updates Nginx to reverse-proxy requests.
   * **Subdomain Logic**: Branch subdomains (branch-name.test.drivemate.au) are dynamically added/removed in Route53.
2. **SSL Configuration:**
   * **Centralized SSL Management:** 
     + AWS ACM provides an easy way to manage SSL certificates, avoiding manual certificate generation and renewal.
   * **Scalability:**
     + Load Balancer handles traffic distribution and HTTPS termination for multiple subdomains.
   * **Security:**
     + ACM ensures secure, managed certificates with encryption and compliance standards.
3. **Cleanup Logic:**
   * **Branch Resource Cleanup**:
     + Remove containers, images, subdomains, and Nginx configurations for unused branches.
   * **Route53 Record Cleanup**:
     + Ensure DNS entries for inactive branches are deleted.

### **AWS Configuration**

1. **Set up AWS Resources:**
   * Launch an EC2 instance with Docker, Nginx, and the necessary IAM role for GitHub Actions.
   * Create an Amazon ECR repository for each branch or environment.
2. **Set Up IAM Role**:
   * Create a role with permissions for:
     + ECR (Push and pull images)
     + EC2 (Security group modifications)
     + Secrets Manager (Retrieve secrets)
     + Route 53 (Manage DNS records)
   * Allow GitHub OIDC provider to assume this role.
3. **Secrets in AWS Secrets Manager**:
   * **Environment Secrets**:
     + Environment-specific variables (e.g., database credentials, API keys).
   * **Deployment Secrets**:
     + EC2 SSH key (Base64 encoded)
     + Route53 zone ID
     + Load Balancer details (ARN, domain, SSL certificate)
     + Security group ID
4. **Security Groups**:
   * Ensure the EC2 instance security group allows incoming traffic on ports specified during container deployment.
5. **Route 53**:
   * Create a hosted zone and update DNS records dynamically for subdomain routing.
   * Configure the domain and subdomains to route to your load balancer or EC2 instance.

### **GitHub Secrets**

1. AWS\_ACCOUNT\_ID: Your AWS account ID.
2. AWS\_REGION: AWS region for deployments.

### **AWS Secrets Manager Keys**

1. **Environment Secrets**

* ENVIRONMENT\_SECRET\_ID:
  + Contains application environment variables (e.g., drivelah-au/web/production).

1. **Deployment Secrets**

* DEPLOYMENT\_SECRET\_ID: Contains sensitive information like:
  + EC2\_USER: SSH user for EC2.
  + EC2\_HOST: Public IP or hostname of the EC2 instance.
  + ENCODED\_EC2\_KEY: Base64-encoded private SSH key.
  + SECURITY\_GROUP\_ID: Security group for the EC2 instance.
  + ROUTE53\_ZONE\_ID: Hosted zone ID for Route 53.
  + BASE\_DOMAIN: Base domain (e.g., test.drivemate.au).
  + LOAD\_BALANCER\_DOMAIN: Domain for the load balancer.
  + SSL\_CERTIFICATE\_ARN: ARN of the SSL certificate for HTTPS.
  + LOAD\_BALANCER\_ARN: ARN of the load balancer for branch routing.
  + TARGET\_GROUP\_ARN: ARN of the target group for load balancer.

### **Pipeline Configuration on AWS**

1. **Create IAM Role**:
   * Name: github-action-deployment-role.
   * Policies: Attach managed policies for ECR, EC2, Secrets Manager, and Route 53.
2. **Configure Secrets Manager**:
   * Add secrets for ENVIRONMENT\_SECRET\_ID and DEPLOYMENT\_SECRET\_ID.
3. **Set Up ECR Repository**:
   * Create a repository and note the repository URI.
4. **Update EC2 Security Groups**:
   * Allow SSH (port 22) and application-specific ports.

### **Pipeline Flow Summary**

1. **Branch Configuration:**
   * Add branch configurations in branches-config.yml.
2. **Build and Push:**
   * Build Docker images and push them to ECR.
3. **Deploy:**
   * Start the Docker container for the branch.
   * Set up a subdomain and configure SSL using Nginx and Route53.
4. **Cleanup:**
   * Stop and remove inactive branch resources (containers, subdomains, Nginx configs, etc.).

### **How to Use the Pipeline**

1. **Add Branch Configurations:**
   * Update the branches-config.yml file for new branches.
2. **Push Code:**
   * Commit and push changes to GitHub.
3. **Monitor Deployments**:
   * Check the deployment status via GitHub Actions and config.test.drivemate.au.
4. **Cleanup:**
   * Ensure unused branches are removed from the configuration file to trigger cleanup.

### **FAQs**

1. **What happens if a branch is not listed in branches-config.yml?**
   * The pipeline stops and exits with a message indicating the missing configuration.
2. **How is zero downtime ensured?**
   * Old containers are removed only after the new container is live and verified.
3. **What happens to outdated branches?**
   * They are cleaned up in the cleanup stage to free resources.