India Launch: Detailed Step-by-Step Plan
(All acronyms expanded and each step elaborated with "what" and "why")

---

1. Prerequisites

1. **Amazon Web Services (AWS) Account & Identity and Access Management (IAM) Permissions**
    - **What:**
        - An AWS account with an IAM user or role that has permissions to create and manage:
            - **App Runner** (fully managed container service)
            - **Amazon Relational Database Service (RDS)**
            - **Virtual Private Cloud (VPC)** and related networking
            - **Amazon Route 53** (DNS service)
            - **AWS Secrets Manager** (credential storage)
    - **Why:**
        - Ensures you have the "right" (least-privilege) access to provision, configure, and secure all necessary cloud resources without using overly permissive credentials.
2. **Domain Hosted in Amazon Route 53**
    - **What:**
        - A registered domain (for example, `yourdomain.com`) with a **hosted zone** configured in Amazon Route 53.
        - You'll use a subdomain like `erp.yourdomain.com` for your ERP's public endpoint.
    - **Why:**
        - Allows you to map a friendly, branded, and secure URL (with HTTPS) to your App Runner service later.
3. **AWS Command Line Interface (CLI) (Optional but Recommended)**
    - **What:**
        - The AWS CLI installed on your local machine or available via AWS CloudShell.
        - Configured with your IAM user's access key and secret key.
    - **Why:**
        - Facilitates scripting and automation of repetitive tasks (e.g., creating VPCs, RDS instances) and enables infrastructure-as-code workflows.

---

2. Network & Security Setup

1. **Virtual Private Cloud (VPC) Selection or Creation**
    - **What:**
        - Use the **default VPC** in the **Asia Pacific (Mumbai)** Region (`ap-south-1`) or create a new VPC with at least two **subnets** spread across different **Availability Zones (AZs)**.
    - **Why:**
        - A VPC provides an isolated network environment.

        - Multiple AZs ensure high availability—if one AZ fails, resources in the other AZ can continue serving traffic.
2. **Security Group for RDS (Database) Access**
    - **What:**
        - Create a **security group** (e.g., named `erp-rds-sg`) within your VPC.
        - Configure **inbound rules** to allow only PostgreSQL traffic (TCP port 5432) from the **App Runner VPC Connector**'s CIDR block (the IP range App Runner will use).
        - Keep **outbound rules** open (default "all traffic allowed") so the database can communicate for backups, monitoring, etc.
    - **Why:**
        - Security groups act as virtual firewalls.
        - Restricting inbound access to only your application prevents unauthorized database connections.
3. **(Optional) VPC Connector for App Runner**
    - **What:**
        - Define an **App Runner VPC Connector** resource that integrates App Runner services into your private VPC network.
    - **Why:**
        - By default, App Runner services run in a managed network.
        - Attaching a VPC Connector gives your containerized application private network access to RDS—no need to expose the database publicly.

---

3. Provision Amazon RDS PostgreSQL (Multi-AZ)

- **What:**
    - In the RDS console, choose **"Create database"** with engine **PostgreSQL**.
    - Select the **db.t3.medium** instance class (2 vCPU, 4 GiB RAM) with **Multi-AZ deployment** enabled in the Mumbai Region (`ap-south-1`).
    - Configure storage: 20 GiB GP3 (General Purpose SSD) with auto-scaling.
    - Set **backup retention** to 7 days and enable **performance insights** if desired.
- **Why:**
    - **Multi-AZ** ensures automatic failover to a standby instance in another AZ if the primary fails—critical for high availability.
    - Daily automated backups let you restore point-in-time.
    - Performance insights help you monitor and tune database load.

---

4. Store Database Credentials Securely

- **What:**
    - In **AWS Secrets Manager**, create a new secret of type **"Credentials for RDS database"**.
    - Enter the master **username**, **password**, and the RDS **endpoint** (hostname).
    - Name it something like `erp/production/db`.
- **Why:**

- Secrets Manager encrypts credentials at rest and manages rotation policies.
- Your application can retrieve them at runtime via IAM-based requests—no hard-coding of sensitive information in code or environment variables.

---

5. Deploy Your Application to AWS App Runner

- **What:**
  1. In the **App Runner** console, click **"Create service."**
  2. Select your **source**:
     - **GitHub repository** (connected via OAuth) with build instructions (e.g. Dockerfile), or
     - **Amazon Elastic Container Registry (ECR)** image URI.
  3. Name the service `erp-frontend-india-launch` and set the **Region** to Mumbai (`ap-south-1`).
  4. Configure **instance size** to **1 vCPU / 2 GiB RAM** and **auto-scaling** (min 1, max 2).
  5. Under **Networking**, attach your **VPC Connector** so the service can talk privately to RDS.
  6. Define **environment variables** (pointing to Secrets Manager ARNs or using parameter names):
     - `DB_HOST` → RDS endpoint
     - `DB_NAME`, `DB_USER`, `DB_PASS` → retrieved at runtime from Secrets Manager
  7. Enable **logs** to CloudWatch and accept default HTTP **health check** settings.
- **Why:**
  - App Runner is a fully managed platform—no servers to patch or scale manually.
  - Built-in auto-scaling and health checks keep your service resilient.
  - Private networking with the VPC Connector secures database traffic.

---

6. Point Your Custom Domain & Enable HTTPS

1. **Add Custom Domain in App Runner**
   - In the App Runner service details, under **"Custom domains,"** add `erp.yourdomain.com`.
   - App Runner automatically provisions an **SSL/TLS certificate** via AWS Certificate Manager.
2. **Create Alias Record in Route 53**
   - In your Route 53 hosted zone, create an **A – IPv4 address** record:
     - **Name:** `erp`
     - **Alias:** Yes → select the App Runner custom domain endpoint.

- **Why:**
  - Provides a user-friendly URL.
  - Automatic certificate provisioning ensures all traffic is encrypted (HTTPS) without manual cert management.

---

7. Validation & Smoke Test

- **What:**
  1. Visit `https://erp.yourdomain.com` in your browser—confirm the app loads.
  2. Through the UI, execute key flows such as:
     - **Create Company** → new tenant setup.
     - **List Accounts** → fetch Chart of Accounts from the database.
  3. Review **CloudWatch Logs** for any errors or timeouts.
- **Why:**
  - Verifies end-to-end connectivity (App Runner → RDS).
  - Catches misconfigurations before handing over to users.

---

8. Monitoring, Backups & Security Hardening

1. **Amazon RDS Monitoring & Backups**
   - Ensure **automated backups** are enabled (7-day retention).
   - Turn on **Enhanced Monitoring** at 5-second granularity.
   - Create **CloudWatch Alarms** for:
     - CPU utilization > 80%
     - Storage used > 75% of allocated
     - Replica lag (if future replica exists) > 1 second
2. **App Runner Metrics**
   - Configure CloudWatch Alarms on:
     - Request count anomalies
     - High error rates (5xx status codes)
     - Latency spikes
3. **IAM & Secrets Manager Practices**
   - **Rotate** the database secret every 30–90 days.
   - Limit the App Runner service role to only:
     - `secretsmanager:GetSecretValue` for the specific secret ARN
     - RDS connect permissions (via security group rules)

- **Why:**
  - Early-warning on resource exhaustion or performance degradation.
  - Least-privilege IAM reduces blast radius if credentials or roles are compromised.

---

9. Next Steps & Future Evolution

1. **Continuous Integration / Continuous Deployment (CI/CD)**
   - Connect your Git repository to App Runner so that pushes to `main` trigger new builds and automatic rollouts.

2. **Staging & Production Environments**
    - Replicate this setup in separate AWS accounts or VPCs.
    - Use parameterized CloudFormation or Terraform stacks with distinct environment variables.
3. **Asia Expansion (Read-Replica & Edge Routing)**
    - When you need low-latency reads in Southeast Asia:
        - Repeat RDS provisioning as a **cross-Region read replica** in Singapore (`ap-east-1`).
        - Deploy a second App Runner service in that region, pointing at the replica endpoint.
        - Enable **AWS Global Accelerator** for intelligent routing to the nearest healthy endpoint.

- **Why:**
    - Phased rollout minimizes initial cost and complexity.
    - Allows you to validate India-only operations before scaling geographically.
    - Ensures global users see sub-200 ms response times as you expand.