**Deploying a react app is a straightforward process as application is deployed as static content. It’s a frontend technology so we do not need to configure anything on the server. But on development server where we are doing code changes, we need to install some prerequisites on server. So, we will divide the complete process in two parts i.e. development server and production server.**

**1. Prerequisites for Development server**

Before starting, ensure the following are set up:

* **Node.js** (v16 or newer) and **npm** or **Yarn** installed.
* A GitHub/Bitbucket repository with your frontend code (Optional depending on development process)

**2. Build the Application**

1. **Install Dependencies** Navigate to the project directory and install the required dependencies:

npm install

1. **Build the Application** Create a production build using the following command:

npm run build

This generates a build directory containing optimized static files for production.

**3. Deploying the Application on Production Server**

**Option 1: Deploy on AWS**

**1. Set Up Amazon S3 for Static Hosting**

1. **Create an S3 Bucket**:
   * Go to the [AWS S3 Console](https://s3.console.aws.amazon.com/).
   * Click **Create Bucket**.
   * Enter a unique bucket name (e.g., my-frontend-app).
   * Choose the region closest to your users.
   * Uncheck **Block All Public Access** for public hosting (you can restrict access later via CloudFront).
   * Click **Create Bucket**.
2. **Upload Files to S3**:
   * Navigate to the bucket and click **Upload**.
   * Upload the contents of the build/ directory.
   * Set all files to public (if not using CloudFront).
3. **Enable Static Website Hosting**:
   * Go to the **Properties** tab of the bucket.
   * Scroll to **Static Website Hosting** and click **Edit**.
   * Enable **Static Website Hosting**.
   * Specify the index document as index.html.
   * Save the changes.

**4. Configure Permissions**

1. Update the **Bucket Policy** to allow public access:
   * Go to the **Permissions** tab of the bucket.
   * Scroll to **Bucket Policy** and click **Edit**.
   * Add the following policy (replace BUCKET\_NAME with your bucket name):

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::BUCKET\_NAME/\*"

}

]

}

1. Save the policy.

**5. Set Up Amazon CloudFront (Optional but Recommended)**

1. **Create a CloudFront Distribution**:
   * Go to the [CloudFront Console](https://console.aws.amazon.com/cloudfront/).
   * Click **Create Distribution**.
   * Choose **Web** and configure:
     + Origin Domain: Your S3 bucket's static website endpoint (e.g., my-frontend-app.s3-website-us-east-1.amazonaws.com).
     + Viewer Protocol Policy: Redirect HTTP to HTTPS.
     + Default Root Object: index.html.
   * Click **Create Distribution**.
2. **Update DNS (Custom Domain)**:
   * If using a custom domain, configure your DNS provider (e.g., Route 53) to point to the CloudFront distribution.
   * Add a CNAME record for the domain.

**6. Test Your Deployment**

* Access your application via the CloudFront URL (e.g., https://d12345abcde.cloudfront.net) or the S3 static website URL.
* Verify the React app is functional.

**Option 2: Deploy on a Custom Server**

1. **Transfer the build directory to the root directory of server:**

scp -r build/ user@your-server:/path/to/deploy

1. **Configure the web server:**

**For NGINX:**

server {

listen 80;

server\_name your-domain.com;

root /path/to/deploy/build;

index index.html;

location / {

try\_files $uri /index.html;

}

}

**For Apache: Enable rewrite rules to serve the React app:**

<Directory /path/to/deploy/build>

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

<IfModule mod\_rewrite.c>

RewriteEngine On

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteRule ^ index.html [QSA,L]

</IfModule>

1. **Restart the web server.**

**4. Environment Variables**

If your application relies on environment variables:

1. Create a .env file in the root of your project with the necessary variables.
2. Access them in React with:

const apiUrl = process.env.REACT\_APP\_API\_URL;

1. Ensure these variables are set in your deployment platform.