Google Cloud Platform Deployment Checklist

☁️ Cloud Run with Automatic Git Deployment

This simplified deployment uses Cloud Run's continuous deployment from GitHub, requiring **minimal manual steps** on GCP.

✨ Why This Approach?

Traditional VMs/Kubernetes: Complex, requires manual scaling, patching, load balancing
Cloud Run + Git: Simple, auto-scales, zero-downtime deployments, pay-per-use

Benefits:
- 🚀 **Push to deploy**: Commit → automatic deployment
- 💰 **Cost-effective**: Pay only when requests are served ($0 when idle)
- 🔒 **Secure**: Automatic HTTPS, managed infrastructure
- 📈 **Auto-scaling**: Handles 1 to 1000s of users automatically
- 🛠️ **No-code GCP**: Everything done through console UI
- ⚡ **Fast**: Global CDN, sub-second response times

Perfect for: Small to medium applications with variable traffic

---

✅ Pre-Deployment Tasks

1. Database Cleanup

- [x] **Data cleared** - All reports, comments, and attachments removed
- [x] **Users preserved** - All user accounts remain intact
- [ ] **Clear uploads directory manually** - Remove files from `/nextjs_space/uploads/` folder
- [ ] **Change default passwords** - Update admin, manager, and employee passwords

To clear uploads:

```
cd /home/ubuntu/hotel_shift_log/nextjs_space/uploads
rm -rf *
```

To change passwords (after deployment):

```
# Generate new hash
node -e "const bcrypt = require('bcryptjs'); con-
sole.log(bcrypt.hashSync('NEW_PASSWORD', 12));"

# Update in database via Cloud SQL console or psql
UPDATE "users" SET password = '$2a$12$HASH...' WHERE username = 'admin';
```

2. Push to Git Repository

```
cd /home/ubuntu/hotel_shift_log
git init
git add .
git commit -m "Initial commit - Production ready"
git branch -M main
git remote add origin https://github.com/YOUR_USERNAME/hotel-shift-log.git
git push -u origin main
```

📋 Simplified Deployment Steps (No-Code GCP)

Step 1: Create Cloud SQL Database (via Console)

Navigate to: Cloud SQL Console (https://console.cloud.google.com/sql)

1. Click **Create Instance** → Choose **PostgreSQL**
2. Configure:
   - **Instance ID**: `hotel-shift-log-db`
   - **Password**: Generate a strong password (save it!)
   - **Database version**: PostgreSQL 14
   - **Region**: Choose closest to your users (e.g., `us-central1`)
   - **Machine type**: Shared Core → 1 vCPU, 1.7 GB
   - **Storage**: 10 GB SSD (auto-increase enabled)
3. Click **Show Configuration Options** → **Connections**:
   - Uncheck “Public IP” (not needed)
   - Check “Private IP” (more secure, Cloud Run will connect via VPC)
4. Click **Create** (takes 5-10 minutes)
5. After creation:
   - Go to **Databases** tab → **Create Database** → Name: `hotel_shift_log`
   - **Copy the Connection Name** (format: `project-id:region:instance-name`)

Step 2: Configure Secrets (via Console)

Navigate to: Secret Manager Console (https://console.cloud.google.com/security/secret-manager)

1. Enable **Secret Manager API** (if prompted)
2. Create secrets:

Secret 1: NEXTAUTH_SECRET
- Click **Create Secret**
- Name: `nextauth-secret`
- Secret value: Generate with `openssl rand -base64 32` or use a password manager
- Click **Create**

Secret 2: SMTP_PASSWORD
- Click **Create Secret**
- Name: `smtp-password`
- For Gmail: Generate App Password (https://myaccount.google.com/apppasswords)
- Click **Create**

Secret 3: DATABASE_URL
- Click **Create Secret**
- Name: `database-url`
- Format: `postgresql://postgres:YOUR_DB_PASSWORD@/hotel_shift_log?host=/cloudsql/YOUR_CONNECTION_NAME`
- Replace `YOUR_DB_PASSWORD` with the Cloud SQL password from Step 1
- Replace `YOUR_CONNECTION_NAME` with the connection name from Step 1
- Click **Create**

Step 3: Deploy to Cloud Run (via Console - No Code!)

Navigate to: Cloud Run Console (https://console.cloud.google.com/run)

1. Click **Create Service**
2. Select **Continuously deploy from a repository (source-based)**
3. Click **Set Up with Cloud Build**:
   - **Repository provider**: GitHub
   - **Authenticate and select repository**: `YOUR_USERNAME/hotel-shift-log`
   - **Branch**: `main`
   - **Build type**: Dockerfile or Buildpack (Cloud Run auto-detects Next.js)
   - **Dockerfile path**: Leave default or specify if you have one
   - Click **Save**
4. Configure service:
   - **Service name**: `hotel-shift-log`
   - **Region**: Same as Cloud SQL (e.g., `us-central1`)
   - **Authentication**: **Allow unauthenticated invocations** (users need to access login page)
   - **CPU allocation**: CPU is always allocated
   - **Autoscaling**: Min instances: `1`, Max instances: `10`
   - **Memory**: `2 GiB`
   - **CPU**: `2`
5. Click **Container, Variables & Secrets, Connections**:

Variables tab - Add these environment variables:
- `NEXTAUTH_URL`: Your Cloud Run URL (you'll update this after first deployment)
- `SMTP_HOST`: `smtp.gmail.com`
- `SMTP_PORT`: `587`
- `SMTP_USER`: `your-email@gmail.com`

Secrets tab - Reference the secrets:
- Click **Reference a Secret** → Select `nextauth-secret` → Expose as `NEXTAUTH_SECRET`
- Click **Reference a Secret** → Select `smtp-password` → Expose as `SMTP_PASSWORD`
- Click **Reference a Secret** → Select `database-url` → Expose as `DATABASE_URL`

Connections tab - Connect to Cloud SQL:
- Click **Add Connection**
- Select `hotel-shift-log-db`

1. Click **Create** (Cloud Build will build from GitHub - takes 5-10 minutes)
2. Once deployed:
   - Copy the Cloud Run URL (e.g., `https://hotel-shift-log-abc123.run.app`)
   - Go back to **Edit & Deploy New Revision**

- Update `NEXTAUTH_URL` variable to the actual Cloud Run URL
- Click **Deploy**

Step 4: Run Database Migrations (One-Time Setup)

Option A: Via Cloud Shell (Easiest)

1. Open Cloud Shell (https://console.cloud.google.com/cloudshell)
2. Clone your repository:
   ```
   bash
   git clone https://github.com/YOUR_USERNAME/hotel-shift-log.git
   cd hotel-shift-log/nextjs_space
   ```
3. Install dependencies:
   ```
   bash
   npm install
   ```
4. Connect to Cloud SQL:
   ```
   bash
   gcloud sql connect hotel-shift-log-db --user=postgres --quiet
   # Enter your database password when prompted
   ```
5. In psql, run:
   ```
   sql
   \c hotel_shift_log
   \q
   ```
6. Set DATABASE_URL and run migrations:
   ```
   bash
   export DATABASE_URL="postgresql://postgres:YOUR_PASSWORD@127.0.0.1:5432/hotel_shift_log?connection_limit=1"
   npx prisma db push
   npx prisma db seed
   ```

Option B: Via Cloud Run Job (More Advanced)

Create a one-time job that runs migrations, then use Cloud Run for your app.

Step 5: Update Deployment URL

After first deployment, update this variable in Cloud Run:
- `NEXTAUTH_URL` → Your actual Cloud Run URL

⚠️ Important Note: File Upload Storage

Current Setup: Files are stored locally in the `uploads/` folder.

Cloud Run Limitation: Cloud Run instances are **ephemeral** - uploaded files will be lost when:
- The app redeploys
- Cloud Run scales down/up instances
- Container restarts

Short-term Solution (for initial testing):
- Set min instances to 1 (already configured above)
- Files will persist during a session but may be lost on redeploy

Long-term Solution (recommended for production):
- Migrate file storage to **Google Cloud Storage**
- Benefits: Persistent, scalable, multi-instance safe
- Implementation: Replace file system writes with Cloud Storage API calls
- Estimated effort: 2-3 hours of development

When to implement Cloud Storage:
- ✅ **Now**: If you expect heavy file uploads or multiple users
- ⏳ **Later**: If you're just testing with a few users initially

---

Step 6: Configure Email Recipients

1. Log into the deployed application as admin
2. Go to **Users** page
3. For each manager who should receive high-priority alerts:
   - Click **Edit**
   - Toggle **Receives High Priority Emails** to ON
   - Ensure email address is filled in
   - Click **Save**

Step 7: Set Up Custom Domain (Optional - via Console)

Navigate to: Your Cloud Run service → **Manage Custom Domains**

1. Click **Add Mapping**
2. Select your Cloud Run service
3. Enter your domain (e.g., `shifts.yourhotel.com`)
4. Follow the DNS verification steps:
   - Add the provided CNAME records to your domain registrar
   - Wait for verification (can take up to 24 hours)
5. Cloud Run will automatically provision SSL certificate

Step 8: Enable Automatic Backups (Already Configured!)

Cloud SQL automatically backs up your database:
- Navigate to your Cloud SQL instance → **Backups** tab
- Verify automated backups are enabled (default: daily at 2 AM)
- Backups are retained for 7 days (increase in **Edit Instance** if needed)

🔒 Security Hardening

Change Default Passwords

- [ ] Admin password changed (username: `admin`)
- [ ] Manager password changed (username: `manager`)
- [ ] Employee password changed (username: `employee`)
- [ ] Cloud SQL password is strong (20+ characters)
- [ ] NEXTAUTH_SECRET is strong (32+ characters)

Configure Monitoring (via Console)

Navigate to: Cloud Monitoring (https://console.cloud.google.com/monitoring)

Set up Uptime Check:
1. Go to **Uptime checks → Create Uptime Check**
2. Configure:
- **Title**: `Hotel Shift Log - Login Page`
- **Protocol**: HTTPS
- **Resource Type**: URL
- **Hostname**: Your Cloud Run URL
- **Path**: `/login`
3. Click **Create**

Set up Error Rate Alert (Optional but Recommended):
1. Go to **Alerting → Create Policy**
2. Add condition:
- **Target**: Cloud Run Revision
- **Metric**: `Request count` (filter for 5xx errors)
- **Threshold**: > 10 errors in 5 minutes
3. Add notification channel (email)
4. Click **Save**

🧪 Post-Deployment Testing

Functionality Tests

- [ ] Can log in with all three roles
- [ ] Employees can create reports
- [ ] Managers can add comments
- [ ] File uploads work correctly
- [ ] High-priority reports send emails
- [ ] PDF/CSV export works
- [ ] User management works (admin only)
- [ ] Archive/unarchive functionality works
- [ ] Report acknowledgement works
- [ ] Comment likes work

Security Tests

- [ ] Cannot access dashboard without login
- [ ] Archived users cannot log in
- [ ] Managers cannot create super admins
- [ ] Employees cannot see manager notes
- [ ] File size limits are enforced
- [ ] Daily post limits are enforced
- [ ] Path traversal blocked (try `../../etc/passwd` in file serving)
- [ ] XSS blocked (try `<script>alert('xss')</script>` in text fields)

Performance Tests

- [ ] Page load time < 3 seconds
- [ ] Large file uploads work (up to 30MB)
- [ ] Can handle 10 concurrent users
- [ ] Database queries are fast

📧 Email Configuration

Gmail Setup (Testing)

1. Enable 2FA on Gmail account
2. Generate App Password: https://myaccount.google.com/apppasswords
3. Use in SMTP_PASSWORD

SendGrid Setup (Production)

1. Create account: https://sendgrid.com/
2. Verify domain
3. Generate API key
4. Configure:
   `SMTP_HOST=smtp.sendgrid.net`
   `SMTP_PORT=587`
   `SMTP_USER=apikey`
   `SMTP_PASSWORD=SG.xxxxxxxxxxx`

Test Email

```
# Create a high-priority report and verify email is received
```

💾 Backup & Restore Procedures

Database Backups (Automatic)

Cloud SQL automatically backs up your database daily. To restore:

1. **Navigate to**: Cloud SQL Console (https://console.cloud.google.com/sql) → `hotel-shift-log-db`
2. Go to **Backups** tab
3. Click on a backup → **Restore**
4. Choose **Restore to same instance** or **Restore to new instance**
5. Confirm (this will overwrite current data if same instance)

Create manual backup before major changes:
- Go to **Backups** tab → **Create Backup**

File Backups (Manual - Only if not using Cloud Storage)

Important: If you're using local file storage, files are **not backed up automatically**.

To back up files:
1. Access your Cloud Run service logs to find where files are stored
2. Download files before major redeployments
3. **Recommended**: Migrate to Cloud Storage for automatic persistence

If using Cloud Storage (future migration):
- Files are automatically versioned and durable
- No manual backup needed

🔄 Continuous Deployment

How it works: Once set up, Cloud Run automatically redeploys when you push to GitHub!

```
# Make code changes locally
git add .
git commit -m "Update feature X"
git push origin main

# Cloud Build automatically:
# 1. Detects the push
# 2. Builds a new container
# 3. Deploys to Cloud Run
# 4. Zero downtime deployment!
```

View build history: Cloud Build Console (https://console.cloud.google.com/cloud-build/builds)

📊 Monitoring & Maintenance

Recommended Alerts (Set up in Cloud Console):

- [ ] Uptime check for `/login` page
- [ ] Error rate > 5% (5xx responses)
- [ ] Response time > 3 seconds
- [ ] Cloud SQL CPU > 80%
- [ ] Cloud SQL storage > 80%

Regular Maintenance:

- **Daily**: Check Cloud Logging for errors (link (https://console.cloud.google.com/logs))
- **Weekly**: Verify database backups are running
- **Monthly**: Review user access and active accounts
- **Quarterly**: Test database restore procedure

🚨 Incident Response

If Unauthorized Access Detected:

1. Immediately revoke all active sessions

2. Change all passwords
3. Review audit logs
4. Enable Cloud SQL read-only mode temporarily
5. Investigate and patch vulnerability
6. Notify affected parties

If Data Loss Occurs:

1. Stop all write operations
2. Identify last known good backup
3. Restore from backup to separate instance
4. Verify data integrity
5. Switch to restored instance
6. Investigate root cause

📝 Documentation Links

- **Full README**: `/home/ubuntu/hotel_shift_log/README.md`
- **Security Analysis**: `/home/ubuntu/hotel_shift_log/SECURITY.md`
- **GCP Console**: https://console.cloud.google.com
- **Cloud SQL**: https://console.cloud.google.com/sql
- **Cloud Run**: https://console.cloud.google.com/run
- **Secret Manager**: https://console.cloud.google.com/security/secret-manager

---

💰 Estimated Monthly Costs

For a small hotel (10-50 users, moderate usage):

Service	Configuration	Est. Cost
Cloud Run	2 GiB RAM, 2 vCPU, min 1 instance	$15-30/month
Cloud SQL	Shared Core, 10GB storage	$10-20/month
Secret Manager	3 secrets	$0.06/month
Cloud Build	120 builds/month	Free (first 120 builds)
Total		**~$25-50/month**

Cost optimization tips:
- Use shared core Cloud SQL initially (upgrade if needed)
- Set min instances to 0 if not mission-critical (saves $10-15/month)
- Monitor with Cloud Billing Reports (https://console.cloud.google.com/billing)

✅ Pre-Launch Checklist

Before Going Live:

- [ ] All default passwords changed
- [ ] Database migrations completed successfully
- [ ] Email notifications tested (high-priority reports)
- [ ] All three user roles tested (admin, manager, employee)
- [ ] File uploads tested (up to 30MB)
- [ ] Uptime monitoring configured
- [ ] Cloud SQL backups verified
- [ ] Team trained on using the system
- [ ] Custom domain configured (if applicable)

Deployment Details:
- **Date**: __
- **Deployed By**: __
- **Cloud Run URL**: __
- **Custom Domain** (if any): __
- **GitHub Repo**: ___

📞 Quick Links & Support

- **Application**: Your Cloud Run URL
- **GCP Console**: https://console.cloud.google.com
- **Cloud Run Dashboard**: https://console.cloud.google.com/run
- **Cloud SQL Dashboard**: https://console.cloud.google.com/sql
- **Logs Viewer**: https://console.cloud.google.com/logs
- **GCP Support**: https://cloud.google.com/support
- **Security Documentation**: `/SECURITY.md`

🎉 You're All Set!

Your Hotel Shift Log application is now running on Google Cloud Platform with:
- ✅ Automatic deployments from GitHub
- ✅ Scalable, managed infrastructure
- ✅ Daily database backups
- ✅ HTTPS encryption
- ✅ Email notifications for high-priority reports

Next Steps:
1. Share the Cloud Run URL with your team
2. Change all default passwords
3. Start using the application!
4. Consider migrating to Cloud Storage for file uploads (when ready)

Document Version: 2.0 (Simplified for Cloud Run + Git)
Last Updated: October 23, 2025