EC2 WordPress Migration Guide
machmacurry.com

This document describes, in detail, the complete end-to-end process followed to migrate the WordPress website **machmacurry.com** from **AWS Account A (EC2-A)** to **AWS Account B (EC2-B)**. It includes the reasoning behind each step, common issues encountered, and the clean best-practice approach used to bring the site live successfully.

1. Migration Objective

The primary goal was to decommission AWS Account A completely while preserving the WordPress website, database, media, themes, plugins, and domain configuration. The website had to be fully operational in AWS Account B with HTTPS enabled and DNS correctly configured.

2. Source Environment (Account A)

- EC2 instance running Ubuntu with Nginx and WordPress (Bitnami-based structure).
- Domain managed via Route 53 in Account A.
- SSL managed using Let’s Encrypt (Certbot).
- WordPress database hosted locally (MySQL/MariaDB).

3. Target Environment (Account B)

- New AWS Account B.
- Fresh EC2 instance (Ubuntu 22.04).
- WordPress deployed manually (non-Bitnami).
- DNS to be managed independently (Route 53 optional).

4. Data Backup from EC2-A

Why: A complete backup ensures that no content is lost during migration.

Actions:
1. Database backup using mysqldump.
2. WordPress files backup including wp-content, themes, plugins, and uploads.
3. Verified backup integrity before transfer.

5. File Transfer to Local Machine

Why: Local storage provides a safe intermediate checkpoint before restoring to EC2-B.

Actions:
- Used SCP/PSCP to download database dump and WordPress tar archive from EC2-A.
- Validated file sizes and permissions locally.

6. EC2-B Instance Setup

Why: A clean environment avoids configuration conflicts from legacy setups.

Actions:
- Launched new EC2 instance using correct key pair.
- Installed Nginx, PHP, MySQL/MariaDB, and required PHP extensions.
- Configured security groups for SSH, HTTP, and HTTPS.

7. Restoring WordPress Files

Why: WordPress content, themes, and plugins reside in the filesystem.

Actions:
- Extracted wp-content into /var/www/wordpress.
- Fixed symbolic links created earlier by Bitnami paths.
- Corrected ownership to www-data:www-data.

8. Restoring Database

Why: WordPress settings, pages, users, and configuration are database-driven.

Actions:
- Created WordPress database and user.
- Imported SQL dump into MySQL.
- Verified tables and row counts.

9. wp-config.php Configuration

Why: WordPress must connect to the correct database and environment.

Actions:
- Updated DB_NAME, DB_USER, DB_PASSWORD, DB_HOST.
- Ensured correct table prefix.
- Fixed broken symlinks and ensured correct file location.

10. Fixing wp-admin and Login Issues

Issues Observed:
- wp-login.php working but wp-admin returning 404.
- Incorrect redirects to localhost or old IP.

Fix:
- Corrected Nginx root and WordPress siteurl/home values.
- Replaced old IP addresses in database.

11. DNS & Domain Migration

Why: Domain must point to Account B after Account A deletion.

Actions:
- Updated GoDaddy nameservers (if using Route 53) or A records directly.
- Pointed machmacurry.com and www.machmacurry.com to new EC2 Elastic IP.

12. SSL (HTTPS) Setup

Why: HTTPS is mandatory for security, SEO, and browser trust.

Actions:
- Installed Certbot.
- Fixed Nginx server_name configuration.
- Issued Let’s Encrypt certificate successfully.
- Enabled automatic renewal.

13. Final Validation

- Homepage loads correctly.
- wp-admin and wp-login work.
- Plugins and themes load without errors.
- HTTPS lock verified.
- No references to old EC2 or Account A remain.

14. Safe Decommissioning of Account A

Why: Avoid unnecessary cost and security risk.

Actions:
- Verified site stability on Account B.
- Terminated EC2-A.
- Deleted Route 53 hosted zone from Account A.
- Closed Account A safely.

Result:
machmacurry.com is now fully operational in AWS Account B with a clean, secure, and maintainable architecture.