Below is a **direct, opinionated plan** for how to manage your [WordPress.com](http://wordpress.com) project **rapidly and without deep DevOps expertise**. This approach keeps things straightforward while still giving you a safe, repeatable process to develop, test, and deploy custom code.

## **1.**

## **Use the** [**WordPress.com**](http://wordpress.com) **Business (or Ecommerce) Plan**

* **Why?** These plans let you install custom plugins/themes, which is essential for your LMS, membership, booking, and any custom functionality.
* Once upgraded, confirm you have access to **SFTP** (or at least the ability to upload custom plugins and themes).

Reset the sapl.academy wordpress and get yourself and Sachin added as Wordpress admins and get that passwords from Khushan/Nikhil

## **2.**

## **Set Up a Local Development Environment**

1. **Local by Flywheel**
   * Download and install [Local](https://localwp.com/) (free).
   * This tool automatically configures PHP, MySQL, and WordPress for you—no advanced server knowledge needed.

Set this up following any standard wordpress tutorial

1. **Match Your Production Setup**
   * In Local, install the **same theme** you plan to use, plus your core plugins (Tutor LMS, MemberPress, Amelia, etc.).
   * This ensures you’re testing code in an environment as close to production as possible.

Theme: <https://wordpress.org/themes/kadence/>  
Plugins:   
1.for production, lets use free plugins   
Tutorlms: <https://wordpress.org/plugins/tutor/>  
WPAMELIA: https://wordpress.org/plugins/ameliabooking/

// when in live Tutolms:<https://tutorlms.com/signin/> , email: [admin@inditech.co.in](mailto:admin@inditech.co.in), Password: use forget password and setup new password, get otp from dev/nikhil that goes to [admin@inditech.co.in](mailto:admin@inditech.co.in)

## **3. Get this thirdstep done by Khushan, -** [**https://www.cloudways.com/blog/wordpress-github/**](https://www.cloudways.com/blog/wordpress-github/) **-** [**https://foreignerds.com/git-wordpress-workflow-for-local-development-to-staging-site/**](https://foreignerds.com/git-wordpress-workflow-for-local-development-to-staging-site/) **//find more**

## **Use GitHub for Version Control**

1. **Create a Private GitHub Repository**
   * Keep your WordPress child theme (if you have one) and **all custom plugin code** in the repo.
   * Each developer clones it locally, makes changes, and commits them.
2. **Simple Branching**
   * Use a single “main” branch for stable code.
   * For new features or bug fixes, create a short-lived feature branch, test locally, then merge back into main once verified.
   * This keeps your code organized without complex workflows.

## **4.**

## **Create a Staging Site on** [**WordPress.com**](http://wordpress.com)

Standard and you can find tutorials for this

1. **Set Up a Second, Private** [**WordPress.com**](http://wordpress.com) **Site**
   * Call it something like [yourproject-staging.wordpress.com](http://yourproject-staging.wordpress.com).
   * Install the **same theme and plugins** so it matches production.
2. **Upload Code to Staging**
   * When your local code is ready, **zip your custom plugin or child theme folder**.
   * Go to the staging site’s dashboard → “Plugins” → “Add New” → “Upload Plugin” (or “Themes” → “Upload Theme”).
   * Activate and test to ensure everything works with your plugins and data.

## **5.**

## **Deploy to Production**

//work with khushan and do this

1. **Package the Final Code**
   * From your main branch in GitHub, download or export the child theme/plugin folder.
   * Zip it up.
2. **Upload via the** [**WP.com**](http://wp.com) **Dashboard**
   * On your **production** site, go to “Plugins” or “Themes” → “Upload.”
   * Activate the updated code.
   * Confirm everything works (membership rules, booking flows, Zoom integration, Razorpay payments, etc.).

*(If your plan includes SFTP, you can also upload files directly to wp-content/plugins or wp-content/themes, but the .zip upload is typically simpler.)*

## **6.**

## **Ongoing Maintenance**

//test it with khushan

1. **Test Plugin Updates in Staging**
   * Whenever Tutor LMS, MemberPress, or Amelia release an update, install it **on the staging site** first.
   * Verify no conflicts or regressions occur.
2. **Regularly Update Production**
   * After confirming staging is stable, install the same updates on production.
3. **Backup & Security**
   * [WordPress.com](http://wordpress.com) handles backups, but confirm how often they occur and how quickly you can restore.
   * Keep **Wordfence** configured for additional security monitoring.
4. **New Features & Bug Fixes**
   * All new custom code starts locally, gets tested in staging, then moves to production once stable.

## **Why This Plan?**

* **Minimal Specialized Expertise**: Local by Flywheel automates the dev environment, and [WordPress.com](http://wordpress.com)’s dashboards handle backups/security.
* **Simple Staging Workflow**: A second [WP.com](http://wp.com) site acts as your “sandbox,” so you can safely test changes before affecting real users.
* **Straightforward Deployment**: Zipping and uploading custom code is easy—no complex DevOps or command-line tools needed.
* **Version Control**: GitHub organizes your code changes without advanced pipelines or merges, reducing confusion.

### **Final Word**

Following this plan, you’ll have a **clear, repeatable process**: develop locally with GitHub, test on a staging [WP.com](http://wp.com) site, then deploy to production with minimal hassle. This is the **most direct** and **user-friendly** method for a WordPress.com-based project that still requires significant custom code.