Deployment Planning: Everything You Need for Success

The key to success for every enterprise project is a well-thought-out plan. One of the biggest mistakes that most teams make is having practices in place that lead to last-minute scrambling. Due to this, team members are likely to miss important steps.

So, to ensure smooth deployment, a step-by-step process for bringing a project's final stage into production is a must. This is called a deployment plan.

In this post, we're going to cover deployment planning and how you can use it to achieve business success in detail.

What Is Deployment Planning?

First, you might be wondering what deployment planning even is. But before we cover that, let's get even more basic and define deployment.

In general, deployment refers to moving an object to a place where some action can be performed on it. **In the case of software development, deployment means making an application ready for delivery.**

Now that we have an idea of what deployment is, let's get on to deployment planning. Deployment planning is a process that takes even the most minor details into consideration. You can check whether the product is intact and bug-free, and you can ensure a smooth customer experience.

So how does the creation of a deployment plan work? First, the project manager creates a deployment plan. After that, the project team reviews it, before deployment. The project schedule allots time for each activity related to the project. And teams can get insight into the

process by referring to the deployment planning framework during deployment.

Things to Consider While Creating a Deployment Plan

Every company has a vision of what they want their work culture to be. And most know that creating a deployment plan will benefit the company immensely. But not all put in the work to develop a sustainable approach for the betterment of the business, nor do they know the key things to consider before making a deployment plan.

So, here is a list of things to consider while creating a deployment plan. Let's take a look.

Cater to Software, Hardware, and Staff Requirements

Software and hardware needs are two of the most important things to consider before you start deployment planning. Besides, it's important to know how many team members can accomplish prescribed tasks within the stipulated time frame. You have to take note of which team member is doing what, too.

The project planner should have a clear idea of who's involved, the skill sets required, and the dedication level needed. The team must upgrade the software in a timely manner and keep the licenses up to date as well. And that's not all. The team members involved also have to cater to any hardware needs prior to deployment planning.

Involve Maximum People

Here's a rule of thumb: **never try deployment planning at an individual level.** Since a lot of people are involved in the deployment phase, it's wise to include them in deployment planning. Most companies tend to keep the experts in the loop and leave out the newbies. That's a mistake.

While experience holds a lot of value, fresh recruits have brand-new skill sets and innovative ideas. Let the employees know their specific role, even if they're working as a team. When you share the workload evenly during deployment planning, you boost success rates.

Follow the Q x A = E Rule

In the Q x A = E rule, *Q* stands for technical quality. After that, *A* stands for cultural acceptance. Finally, *E* stands for effectiveness. Only with the perfect combo of technical quality and cultural acceptance can a company succeed in deployment planning.

Your team needs to put their heads together to make the product technically sound and innovative. And remember that the end users may not be tech geeks. Keep things simple. Also, you'll want to maintain optimum product quality. It will help your team to successfully plan a deployment.

Consider Why, Where, and How in Order to Choose a Deployment Approach

So now you're ready to pick out your deployment approach. First, you need to articulate why you want sustainable performance for your company. Ask yourself, "How will it lead to the improvement of the business?" Secondly, ask, "Where do I want to see the company in the future?" And of course, you need to plan out the steps to achieve this target. This also refers to how you should begin creating your plan.

There are two approaches to starting your deployment plan.

- The first approach involves rolling out the entire deployment plan at once. Training for the novice and the experienced takes place side by side.

- The second approach confines the new plan to a small area of the business. For instance, you might have a dedicated department to carry out deployments.

The first approach offers greater sustainability since no teams or their members are treated with favoritism. On the other hand, in the second approach, the department responsible might not get support from other areas of the business. Thus, even if it's less risky, this method isn't sustainable.

What Is a Deployment Pipeline?

In software engineering, a pipeline is a group of automated processes. It helps a developer in compiling, building, and deploying code. So, what is a deployment pipeline? It's a similar automated process. All you need to do is get the code from version control and deliver it to your stakeholders.

How to Carry Out Deployment Planning

Having a checklist that mentions tasks is a must for deployment planning. It doesn't matter whether the task is a small update or a large rollout. If migration or deployment goes wrong, it can impact the entire project.

Often, deployment planning can get a little complicated. So, to deal with any potential issues, let's take a look at this software release checklist to flawlessly plan your deployment.

Have Development and Operations Collaborate

For a hassle-free deployment planning, development and operation have to go hand in hand. The operations team should be aware of every important aspect of development. Thus, you'll want to make sure that everyone in the company is on the same page.

If your company isn't already practicing DevOps, at least take a page out of the DevOps playbook. In DevOps, both development and operation teams work together and figure out what's happening. And it's not only that; they also both need to know when a particular task is in action. Because of these practices, they should have an idea of the expected outcomes. For instance, by following this approach, both teams will be aware of early warning signs. They can then carry out the appropriate troubleshooting measures.

Test Frequently

Testing a build in all possible ways is a must for a product to live up to expectations. Let teams perform testing once again just before deployment. Testing the new code on different environments and test versions of hardware minimizes the chance of defects.

Develop a Release Strategy

Are you rolling all the features out at once or in pieces? Do users need to manually update something or make any changes from their side? Before you begin deployment, you must know the answers to all these questions. It helps to implement parallel operations. Then, the team can come up with an effective strategy.

Minimize Changes and Have a Rollback Plan

Too many sudden changes are usually what's responsible for a misbehaving application. Instead of bombarding an upgraded version with every change you want to make, add small changes with each release. That way, it's easier for the team to search for the source of an issue.

Here's something of equal importance: if problems arise in the live environment, have a rollback plan. This means, in the case of issues that are live, you should have the option to return to the most stable product version.

Automate as Much as Possible

Deployment automation is a component of the deployment pipeline. It reduces the likelihood of errors. Not only that, but it helps in the deployment of a few changes at a time instead of all at once. Deployment automation also aids in pushing small changes in the production environment. In the meantime, the developers can focus on unit testing and pushing the changes on to the master build.

If you want to know more about automated deployment, take a look at Plutora's tool for automated deployment planning.

Set Metrics to Monitor Software Performance

Make KPIs monitor CPU usage, page loading time, and other factors at the surface level. Setting up metrics helps in finding the efficiency of different aspects involved in a DevOps toolchain. You can also create custom KPIs that know the performance of the software before and after deployment.

Communicate Before Deployment

Don't just roll out everything at once; give a warning. Any sudden change can create confusion. So the team has to be prepared to address that confusion if the need arises. Communication is important before deployment.

Closely Monitor Performance After Deployment

Set up ways to analyze the app's performance post-deployment. With the help of KPIs, you can compare the software performance with pre-deployment metrics.

Summing It Up

Having a documented deployment plan prevents missing out on important stuff at every step. The key to successful deployment planning is to have a checklist and cross items off as you accomplish them. Just as important is to make sure that all the team members are on the same page. Effectively communicate with each team member.

The conclusion? Deployment planning is a must-have to prevent unforeseen situations and delays.

Learn more about software delivery on www.plutora.com