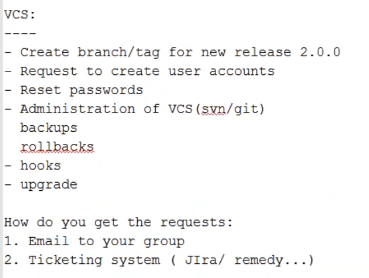


* As above, the release versions would be. First digit for major release, second one for minor & third one is for hotfix
* If we see any number at third place, we can say it is hotfix
* And now the first digit is for internal build number. This number will be keep increasing to get deployed in QA environment
* Once this all done, it would be removed, and only 3-digit version will be deployed to PROD for release

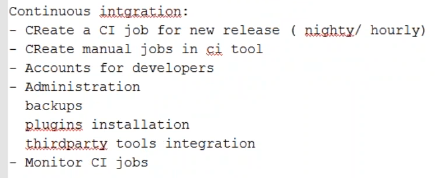
**VCS:**

* In real time, we might be assigned for VCS. There we will get request as below



**Continuous Integration (CI):**

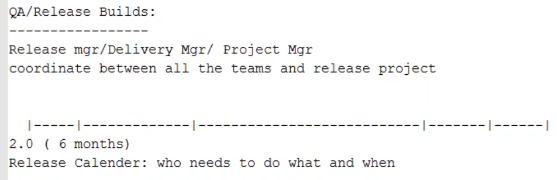
* Whenever we create a new branch, we need to create a CI job as well for new release



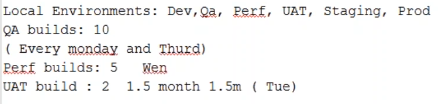
* Monitoring CI jobs is manual. Example, if any nightly job scheduled, we need to monitor in the morning, if it was failed. Then it is our responsibility to escalate the same to developers
* If it is getting failed on the second day also, we might need to escalate to their managers
* We will get instructions to whom we should escalate it. Our job is to monitor it and report to the developers
* These kinds of work are monitoring

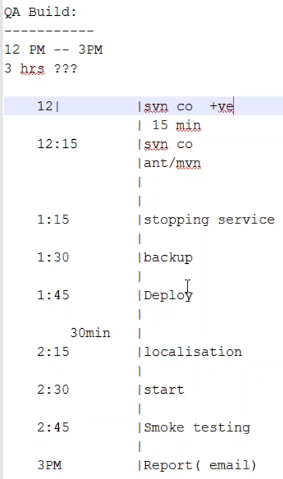
**QA/Release builds:**

* This is our actual responsibility

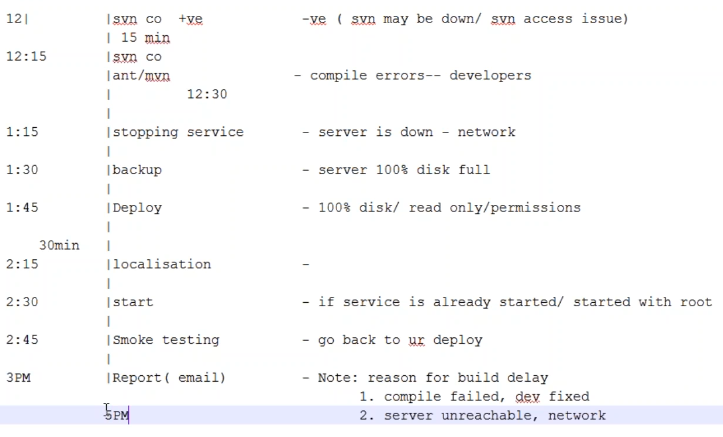


* release manager actually co-ordinates with all the teams like requirement analysis, design, development, testing, production
* his job is to take of all these, getting the requests and assigning to the proper teams. he monitors and if the process is blocked somewhere, he tris to fix them and make the process go smoothly
* manager defines the release calendar and assigns the task to each team
* once requirement analysis and design are done, then the development phase will start
* we work along with developers. but our work will start something like after 10% development done





* if QA build scheduled as above for 3 hrs, our work will be as above for the deployment

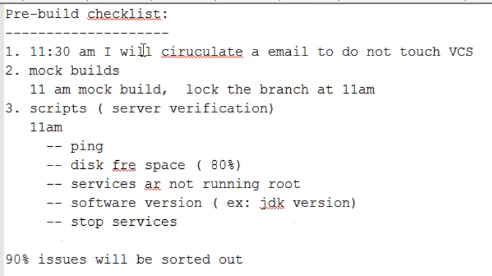


there might me some issues we get as above

1. we may get issue while checking out the code
2. compilation errors, developers will solve those
3. while stopping the service, we won’t get much issues here. if server is not reachable, then network team will solve that
4. while taking backup or deploying, we might face disk space issues. then we need to clear the data and do it. we also might face permissions issue
5. while doing some test, if any issue. we need to check the logs and fix the issue. here it may take lot of time to find and solve the issue
6. once we are all done, we can send an email along with the reasons of delay
7. smoke test also deals with developers also

**pre-build checklist:**

* before build, we can ask the team to not touch the VCS
* we can do the mock the build and lock the branch. so that no one can commit the code
* in some companies, they won’t allow mock build. so, if we want to have the successful build. we better have mock build
* and also, we can write a script lo check the ping of server and to check the disk space and also the to check if the service is running on root account. because if it is running in root account we can not stop or start the service
* we can also write a script to check the jdk version. as sometimes the deployment fails due to the version of JDK
* if we could do all these check prior to build, we can almost sort out 90 % of issues



* we can write the scripts for general purpose like to monitor our services like tomcat. and also, we can write the script to bring the services up if it is down