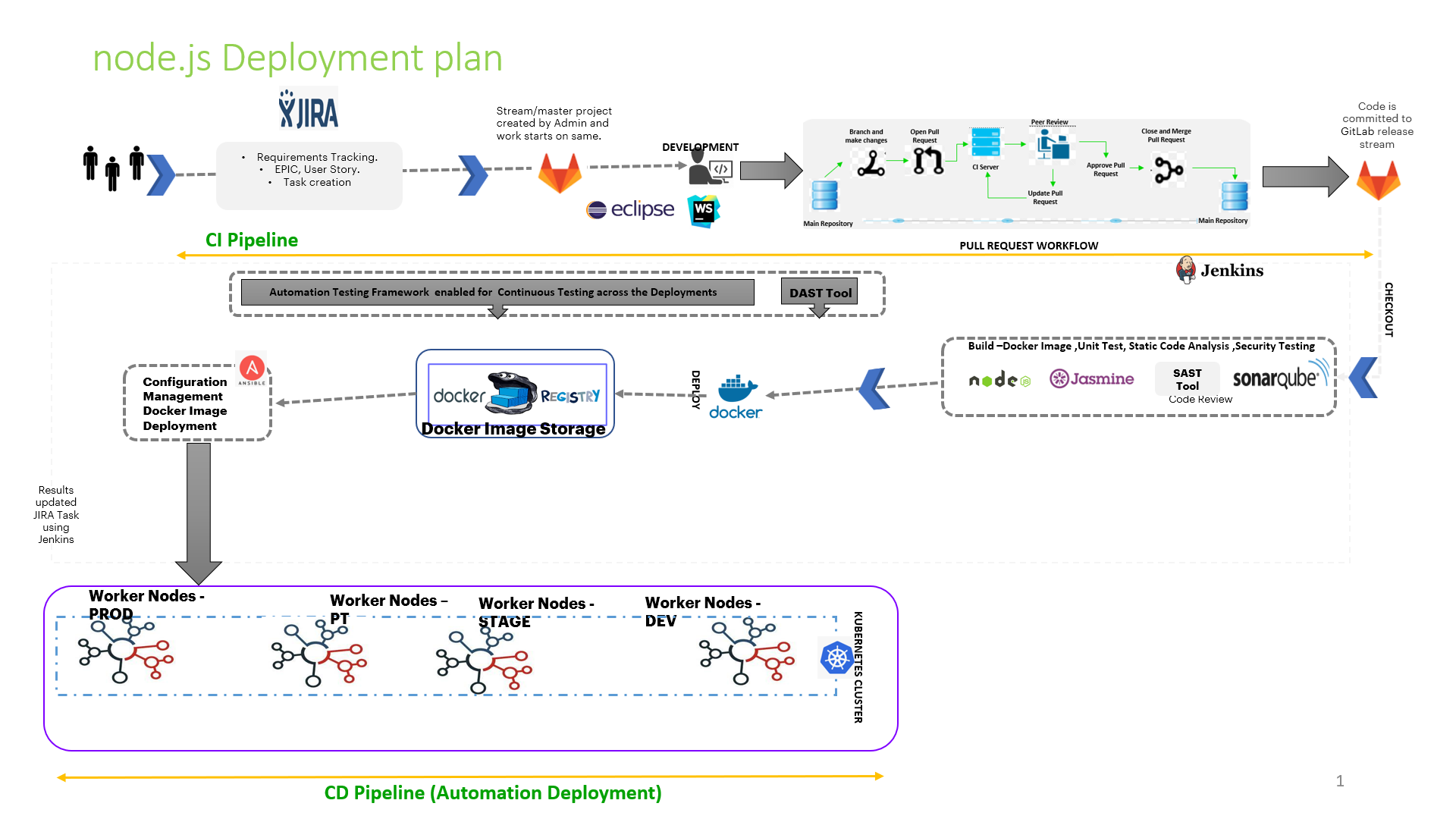
**Node Deployment Plan**

The process involves .js code automated build and deploy to Kubernetes cluster passing through different stages in Jenkins as pipeline as a code.

* Requirements gathering
* Tasks assigned to Developers
* Code developed
* Code checkout from Gitlab
* Unit test cases: Jasmin | Lint (Optional)
* Sonar Analysis (Optional)
* SAST; Static analysis security testing (Optional)
* Yarn build or npm build
* Upload to Nexus (for Backup)
* Deploy to Kubernetes (Different environments as clusters)



This is a high-level blue print of continuous integration and continuous deployment of yarn builds

To start the process from Code check-in by developers, they follow a Unique strategy.

**Git Branching**

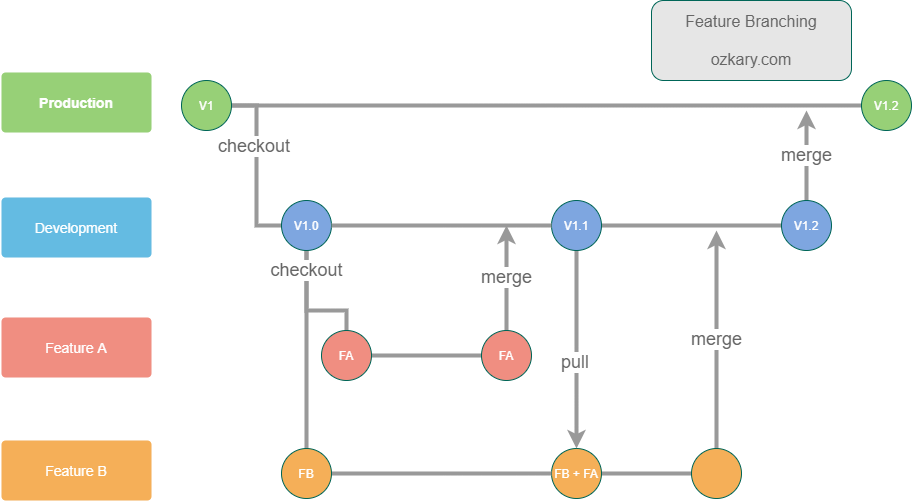
Having a branching strategy is necessary to avoid conflicts when merging and to allow for the easier integration of changes.

The main idea behind the Git branching strategy is to isolate your work into different types of branches.

The branching strategy consists of the following branches:

* Master – Main code of the repository
* Develop – Copy of master but can be developed with new requirements and must be merged back to master.
* Feature - to develop new features that branches off from develop or QA/test branches
* Release - help prepare a new production release; usually branched from the develop branch and must be merged back to both develop and master
* Hotfix - also helps prepare for a release but unlike release branches, hotfix branches arise from a bug that has been discovered and must be resolved; it enables developers to keep working on their own changes on the develop branch while the bug is being fixed.

Branching Strategy



Master

**Whole workflow**

1. Developers gather the requirements mentioned in the Jira ticket and work on the tasks mentioned in there.
2. They will Develop their code in any integrated Development environment.
3. After the development, they will raise the Pull Request to merge their branch with Release branch.
   1. From the requirements, Developers create a branch named feature postfixed with the requirement.
   2. Once the development is done, they will raise a PR from Gitlab, which will trigger an email to the approval group.
   3. Lead group or leads review the code, and update the developers if the code needs to be modified. Else they will approve the PR which will merge that branch to release branch.
   4. Sometimes Leads will forward the PR to Dual review to get a detailed speculation.
4. After the PR got approved, the branches get merged and that will trigger the CI-CD pipelines.
5. Once the pipeline got triggered, it will proceed through below flow:
6. The pipeline will checkout the code from GitLab to Jenkins server.
7. Then the pipeline will run unit test cases on the code.
8. Next, the sonar analysis will be done after the successful execution of unit test cases.
9. After the Sonar success, SAST will run on the code and generate a report and will send the report only if it has critical issues. If no critical issues are found, then it goes to the next stage without notifying.
10. Then the code will get build using Node or Yarn.
11. Then the package will be published to Nexus for versioning as well as for backup.
12. Then Dockerfile, Deployment.yaml and respective certificates will copied into one of the VM
13. From the Dockerfile, the pipeline will pull the respective package from Nexus and build an image along with certificates if needed.
14. That Image will get pushed to Registry for storing and tagging.
15. The Deployment.yaml file will get updated with the image tag that build in the previous step and will get deployed in respective environment.

Additional Steps:

1. User access: Set up user access control to the CI server to ensure that only authorized users can perform builds and deployments.
2. Test: Test the CI server setup to ensure that it is working as expected.
3. Deployment: Deploy the CI server to the production environment, ensuring that it is available 24/7 to perform builds and deployments.
4. Monitoring: Set up monitoring and logging for the CI server to ensure that it is functioning properly and to diagnose any issues that may arise.