DevOps: Activity 1

Customer has an application with high traffic of constant users, and his principal core is sell of things 24/7, product's team frees releases and features new each week on Tuesday.
You have to Developing, or designer, a deployment strategy, and you have to select or create a branch strategy that comply and ensured Real traffic testing and roll back duration fast with these features.

1. You have to that providing an environment for that QA's team can begin test-unit, regression test and stress test of our release and integration API.
2. Likewise, you must guarantee the automatic deployment of application in QA's environment.
3. Furthermore, you must guarantee in application code, good practices of developing, and Measure to efficiency in all status the of developing process.
4. QA team must have autonomy and decision for decliner a feature or release without to affect our business core, nor users of our application.
5. For deploys at production, the confirmation of some is necessary like product owner, tech lead, head's area, or own customer.
6. Releases and features have to be testing in a percentage little of customers of our application without to affected production status.
7. You have to ensure that releases new arrived at production without possible bugs, and then you have to ensure good practices within of the application.

You must create a diagram and support the reason for use of each thing that you represent in the diagram.
Remember, you must use Branch strategies and deployment strategies.

<u>Desarrollo:</u>
Para esta actividad procedimos a realizar de a pares.
Con mi compañero llegamos a la conclusión de que la estrategia de branch mas apropiada para este escenario sería la de Git Workflow debido que es de suma importancia tener la rama de desarrollo (develop), de releases y de producción (main) separadas ya que si hay alguna falla, esta no afecte a la aplicación final, además de que es necesario que los cambios de desarrollo sean testeados y chequeados antes de pasarlos a producción, por lo tanto si la rama de desarrollo tiene una falla, esta no va a afectar a lo que está en producción, ni a la de releases. Entonces cuando todo sea chequeado en desarrollo, recién es unido en la rama de release.
También es importante tener la rama de features para que cada una sea controlada de forma separada y no se mezclen con los demás cambios que se están realizando, y luego de eso se uniría a la rama de desarrollo para ser testeado.

Si se encuentra algún error en producción, este es llevado a ser reparado en la rama de hotfix, y una vez solucionado pasa a la de release para que nuevamente sea chequeado y testeado.
Por otro lado, la estrategia de deploy elegida es la Canary para que las releases sean testeadas en un porcentaje de usuarios sin afectar a lo que está en producción