*Facultad de Ingeniería*

*Ingenieria de Software II*

**Laboratorio 4**

***Realizado por:***

*Juan Pablo Betancourt Maldonado -* [*jpbetancourtma@unal.edu.co*](mailto:jpbetancourtma@unal.edu.co)

*Jaime Eduardo Estupiñan -* [*jestupinanb@unal.edu.co*](mailto:jestupinanb@unal.edu.co)

*Tom Erick Perez Alvarez - tpereza@unal.edu.co*

***Profesor:***

*Yury Yineth Nino Roa*

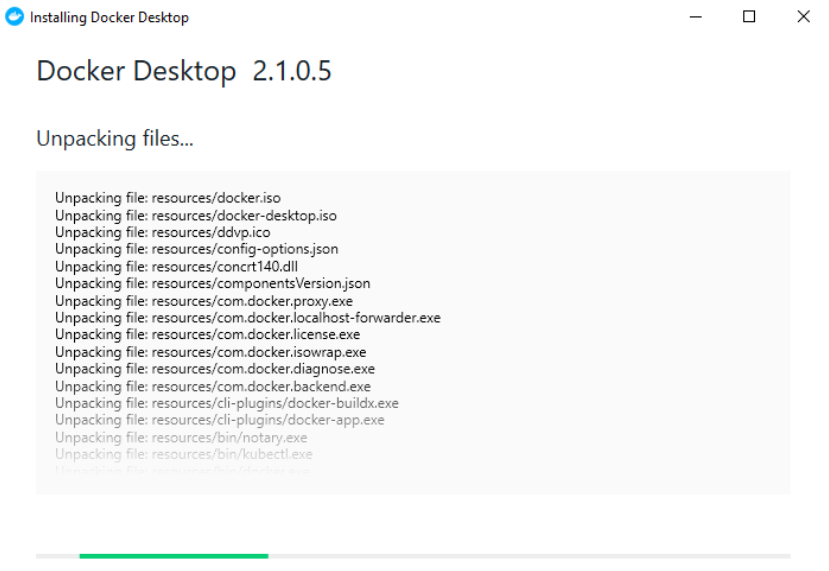
**

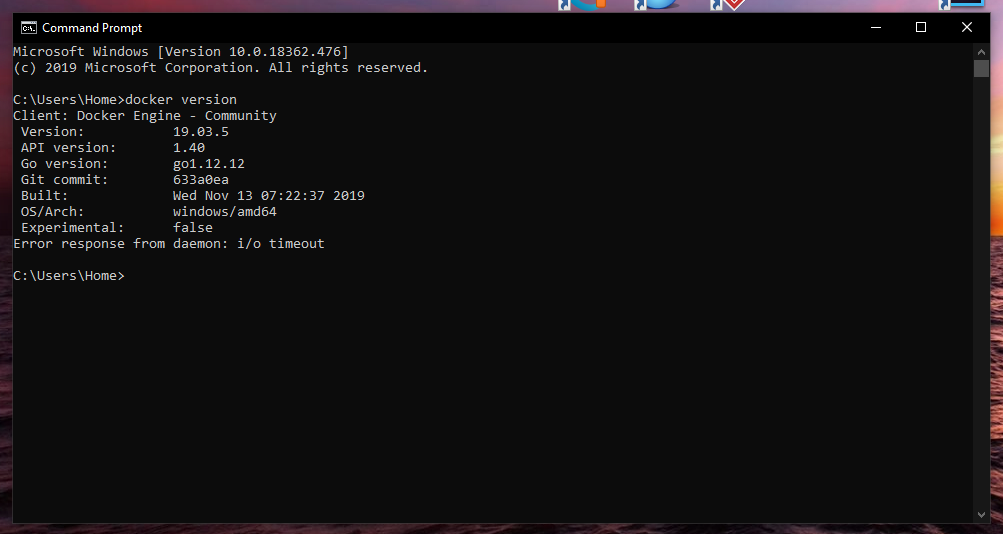
*Cundinamarca*

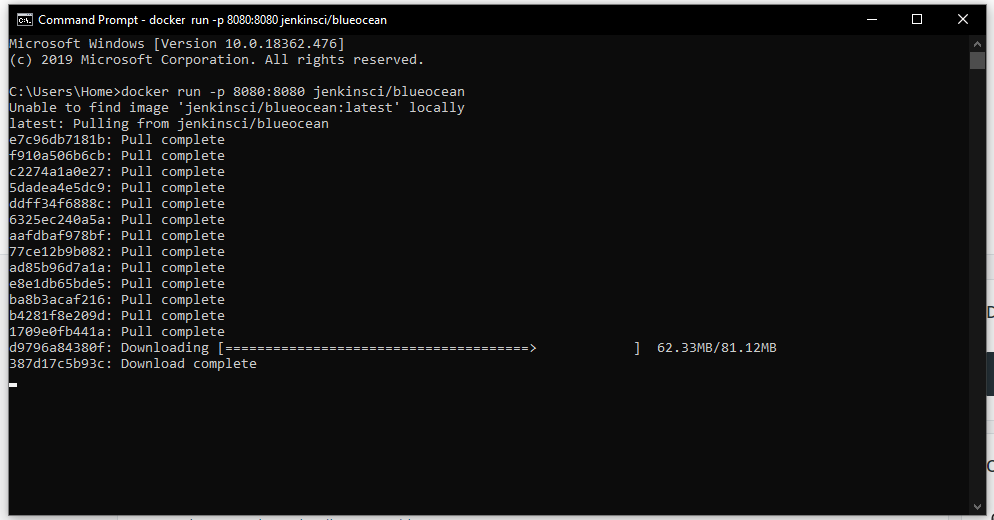
*Bogotá, Colombia*

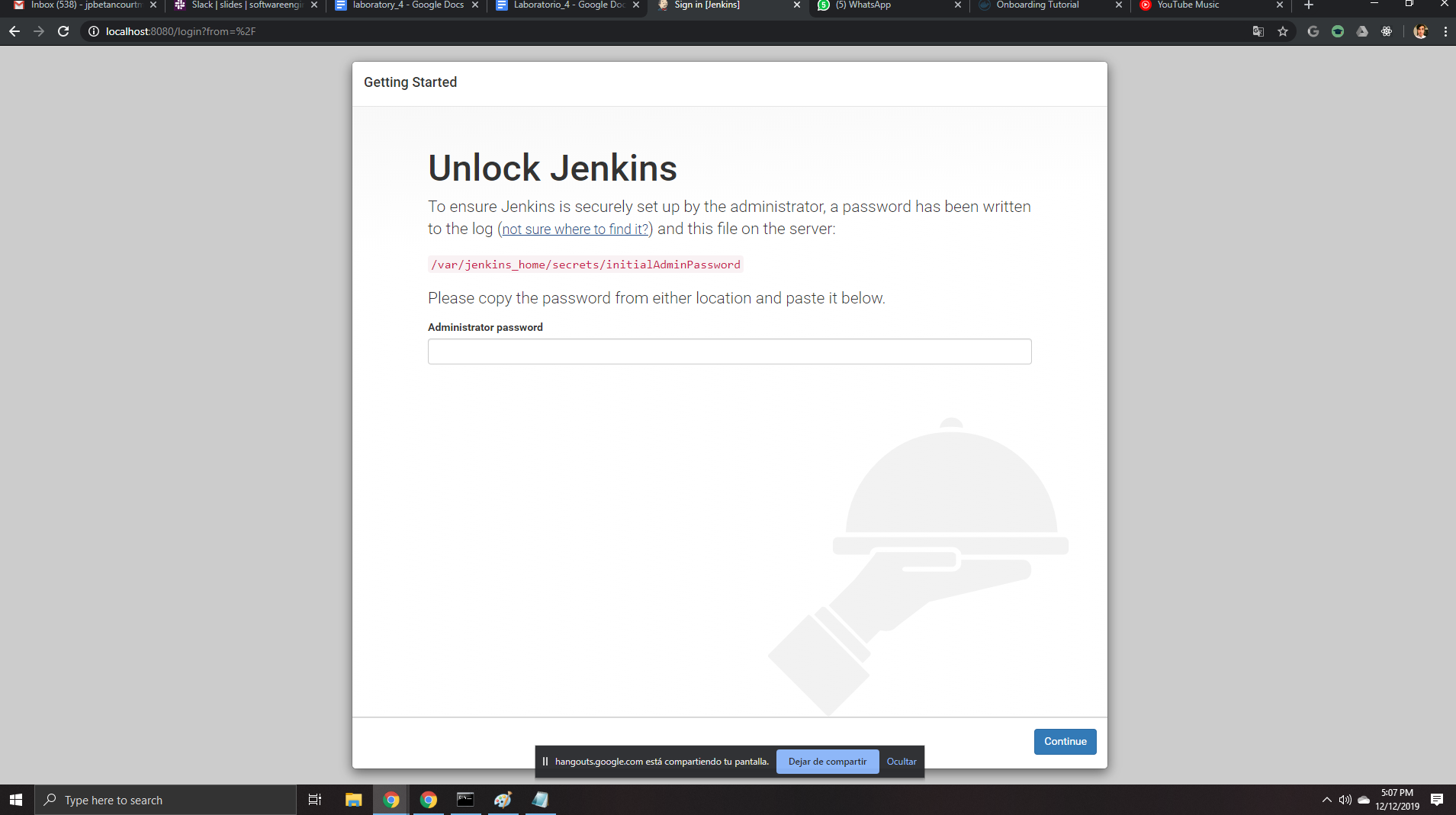
*12 de Diciembre de 2019*

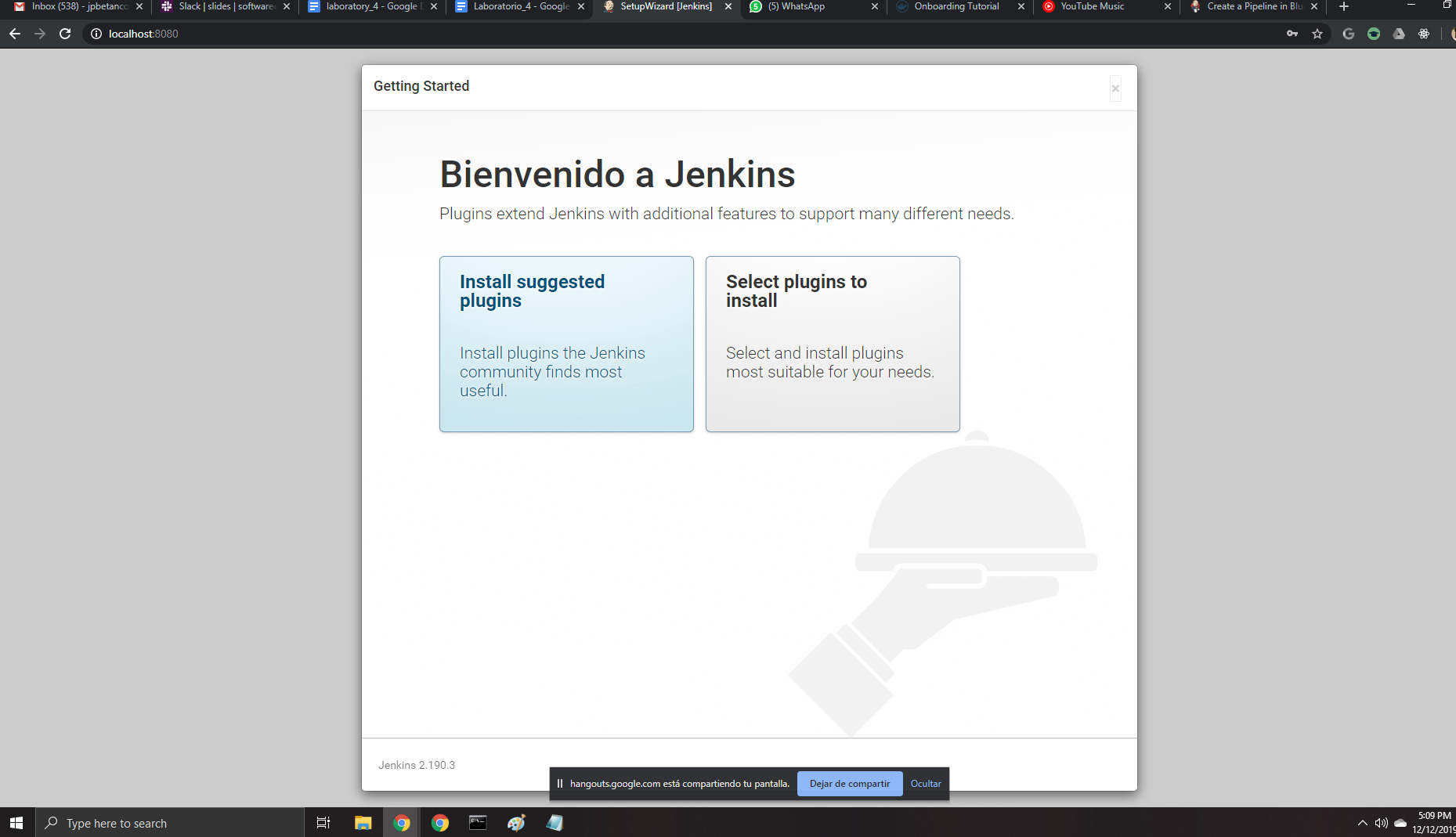
**1 - Install the plugin blue ocean following these instructions** [**https://jenkins.io/doc/book/blueocean/getting-started/**](https://jenkins.io/doc/book/blueocean/getting-started/)**.**

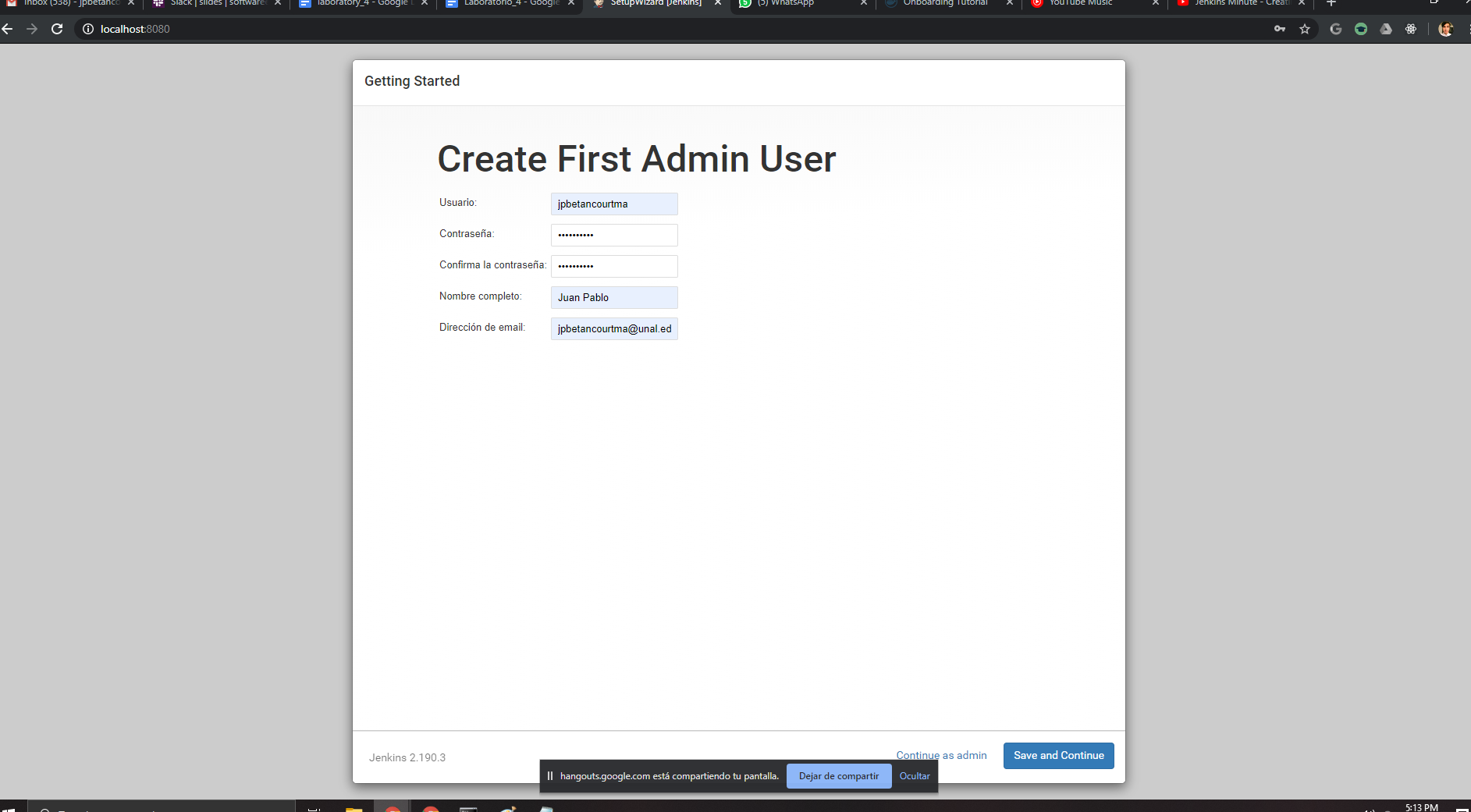


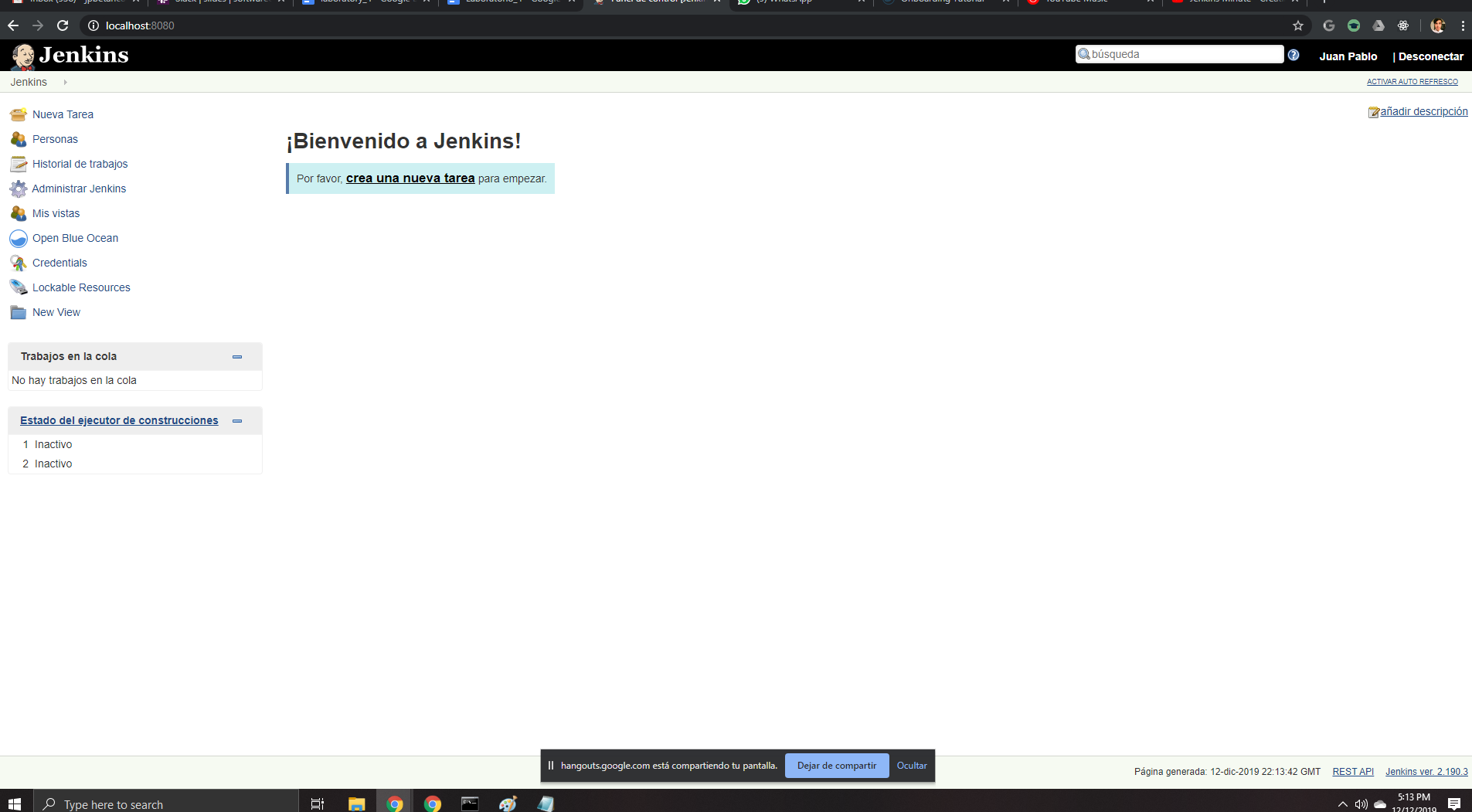


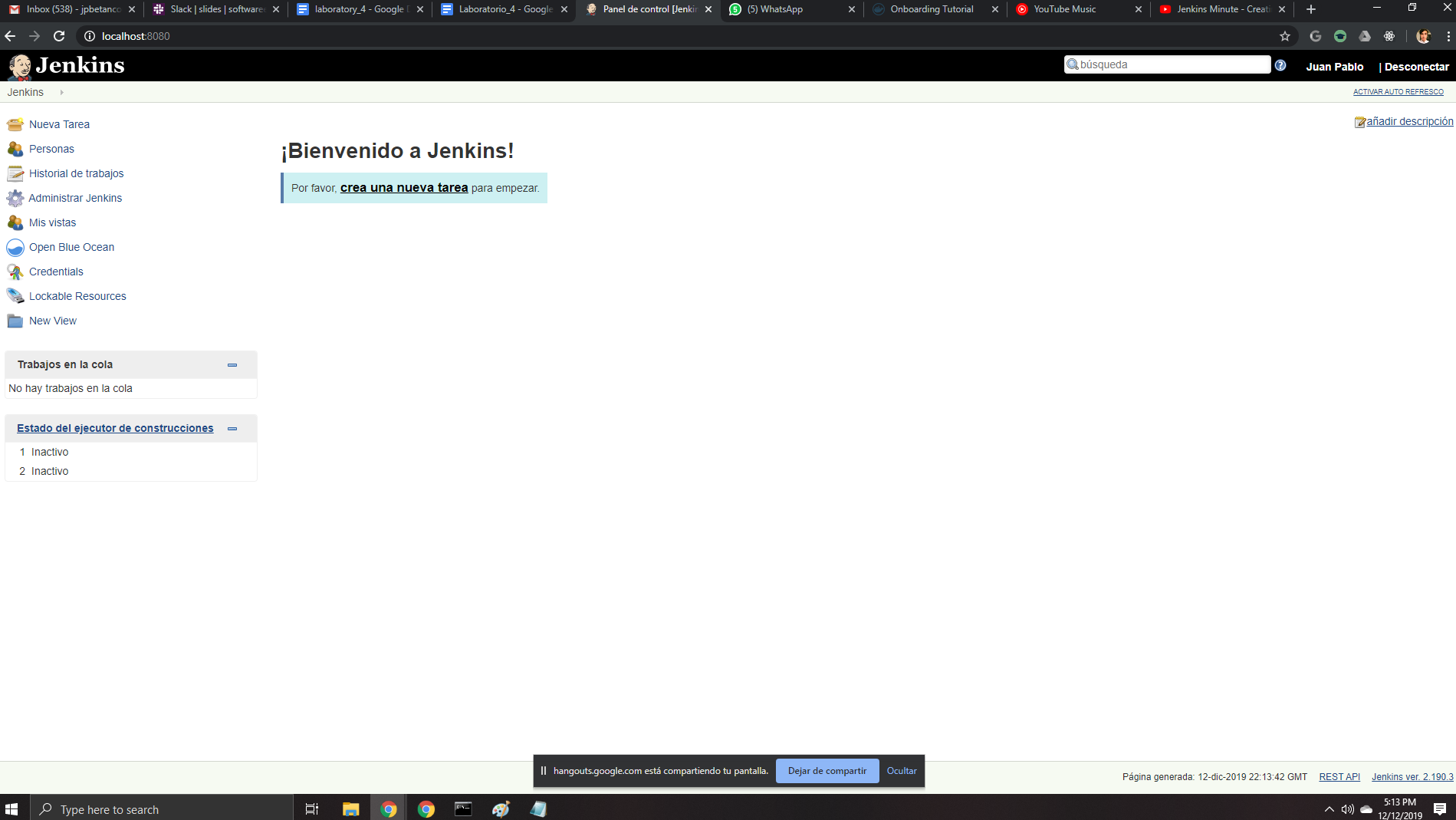


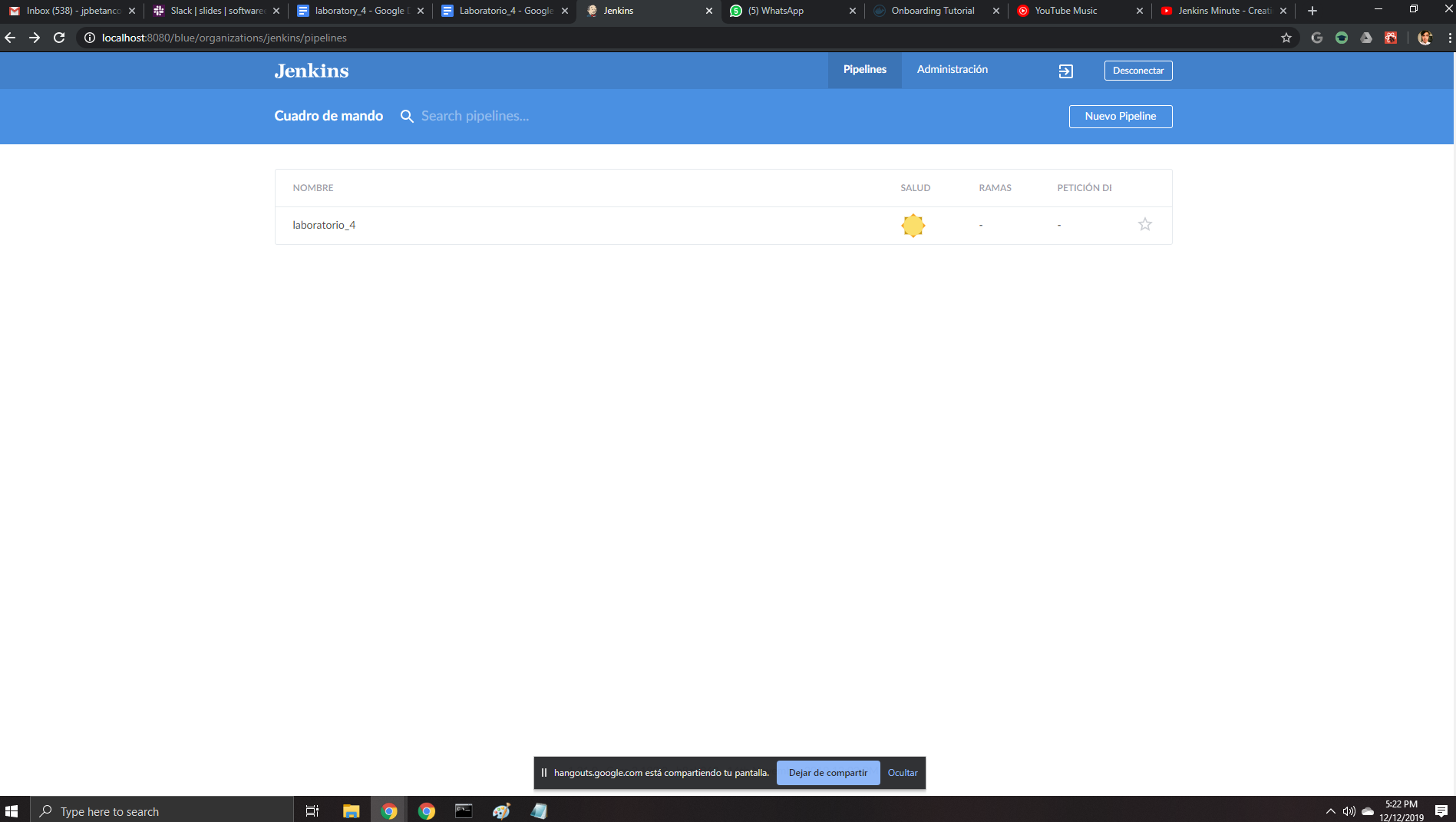




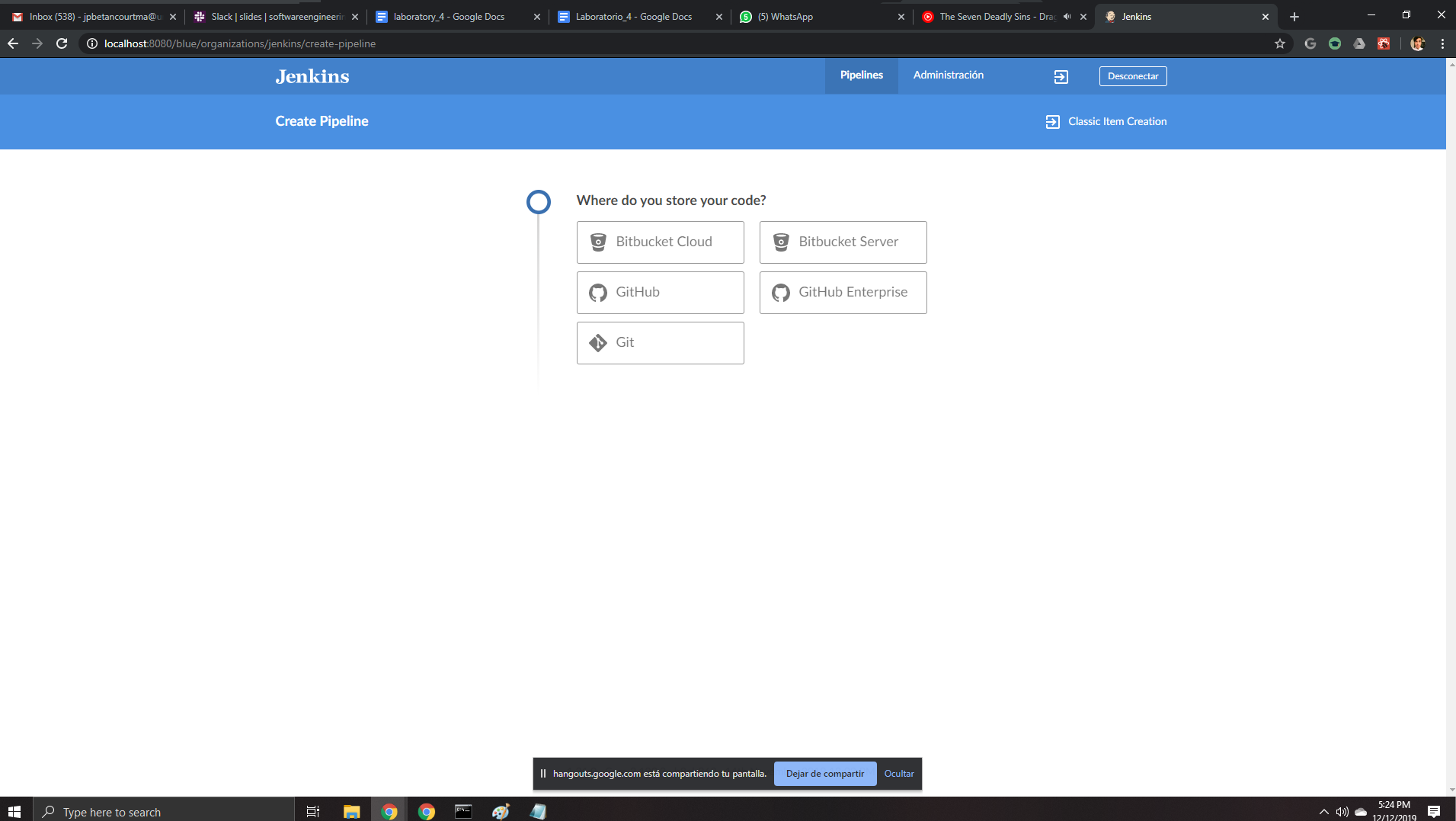


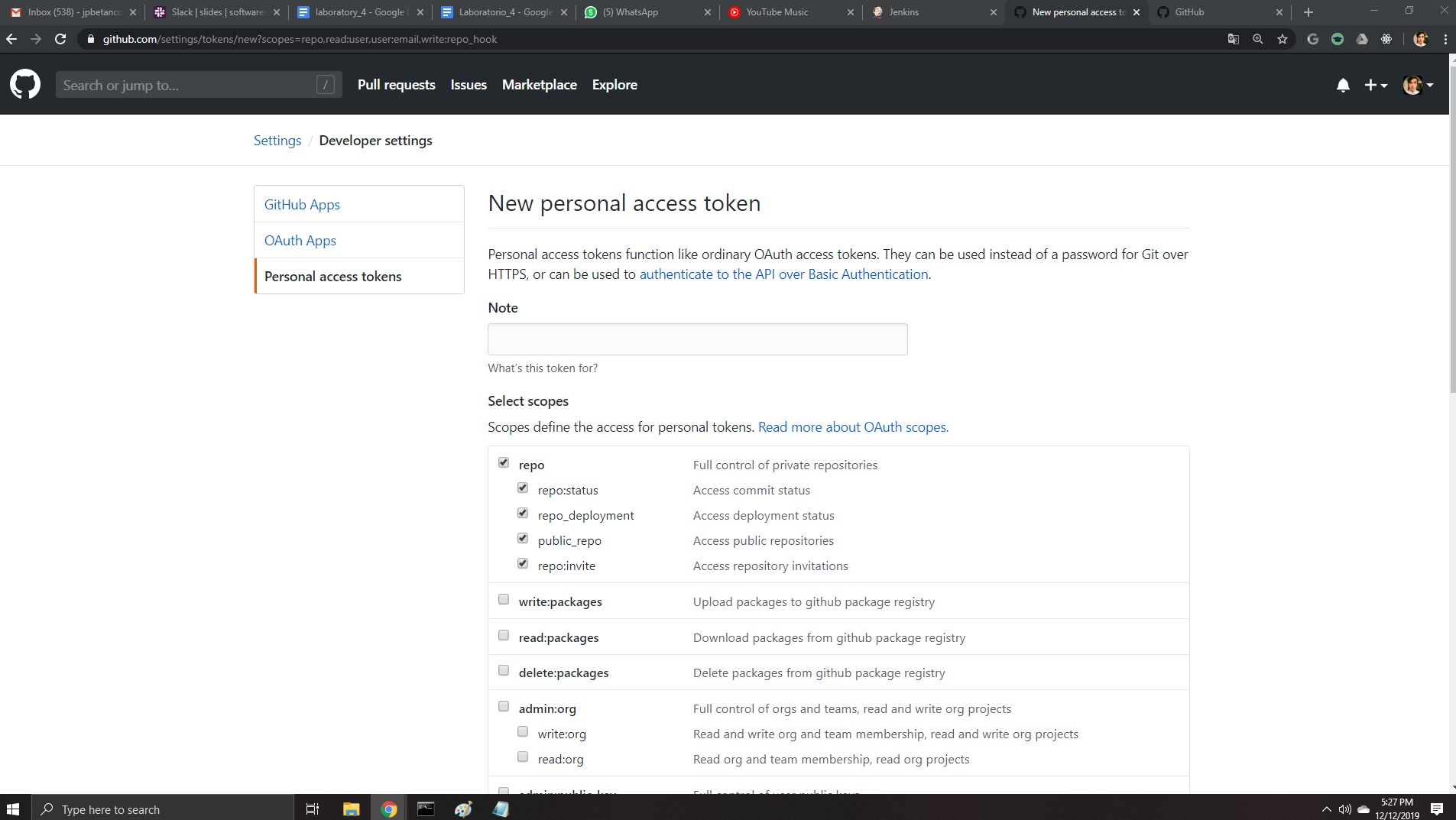


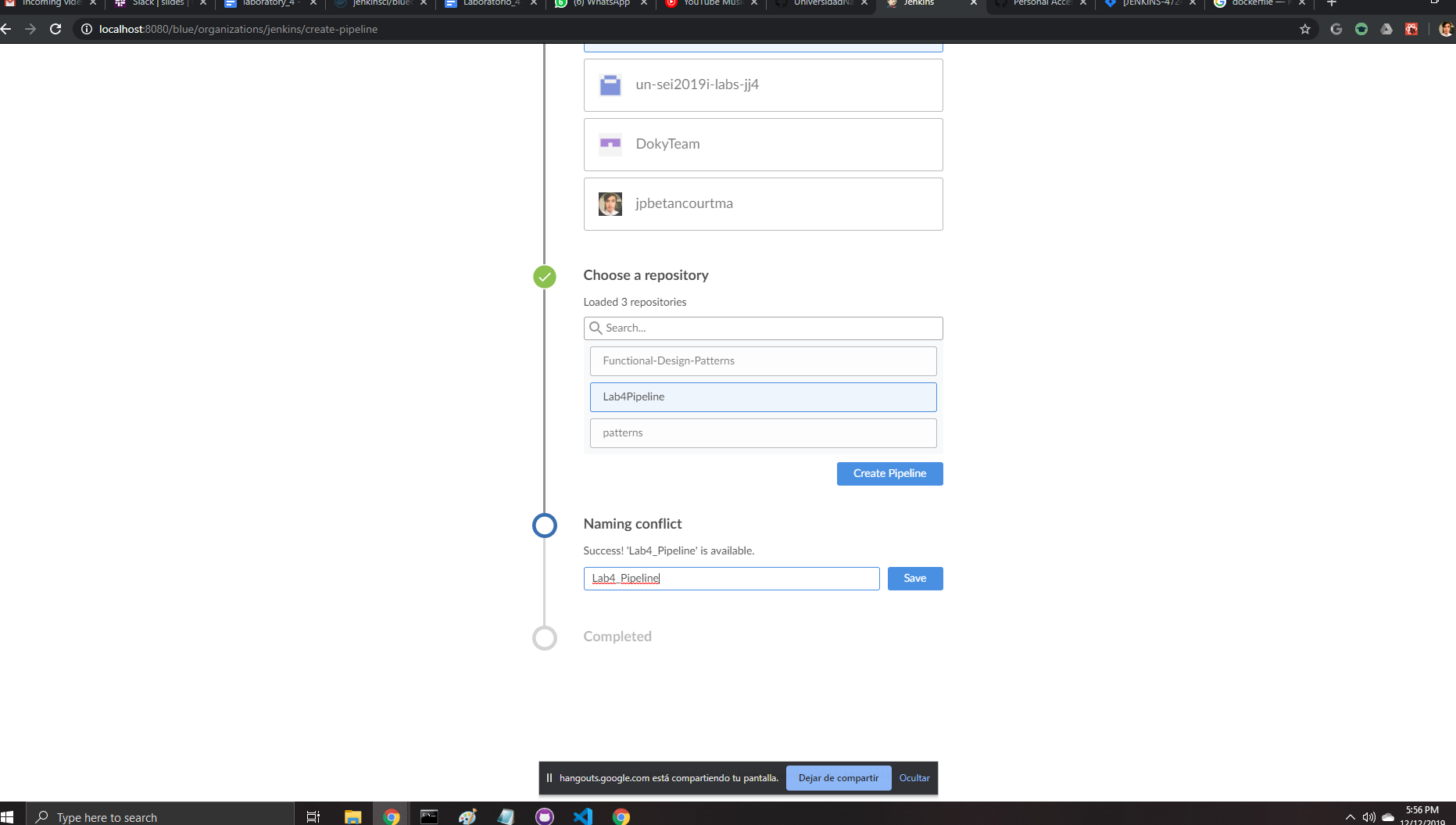


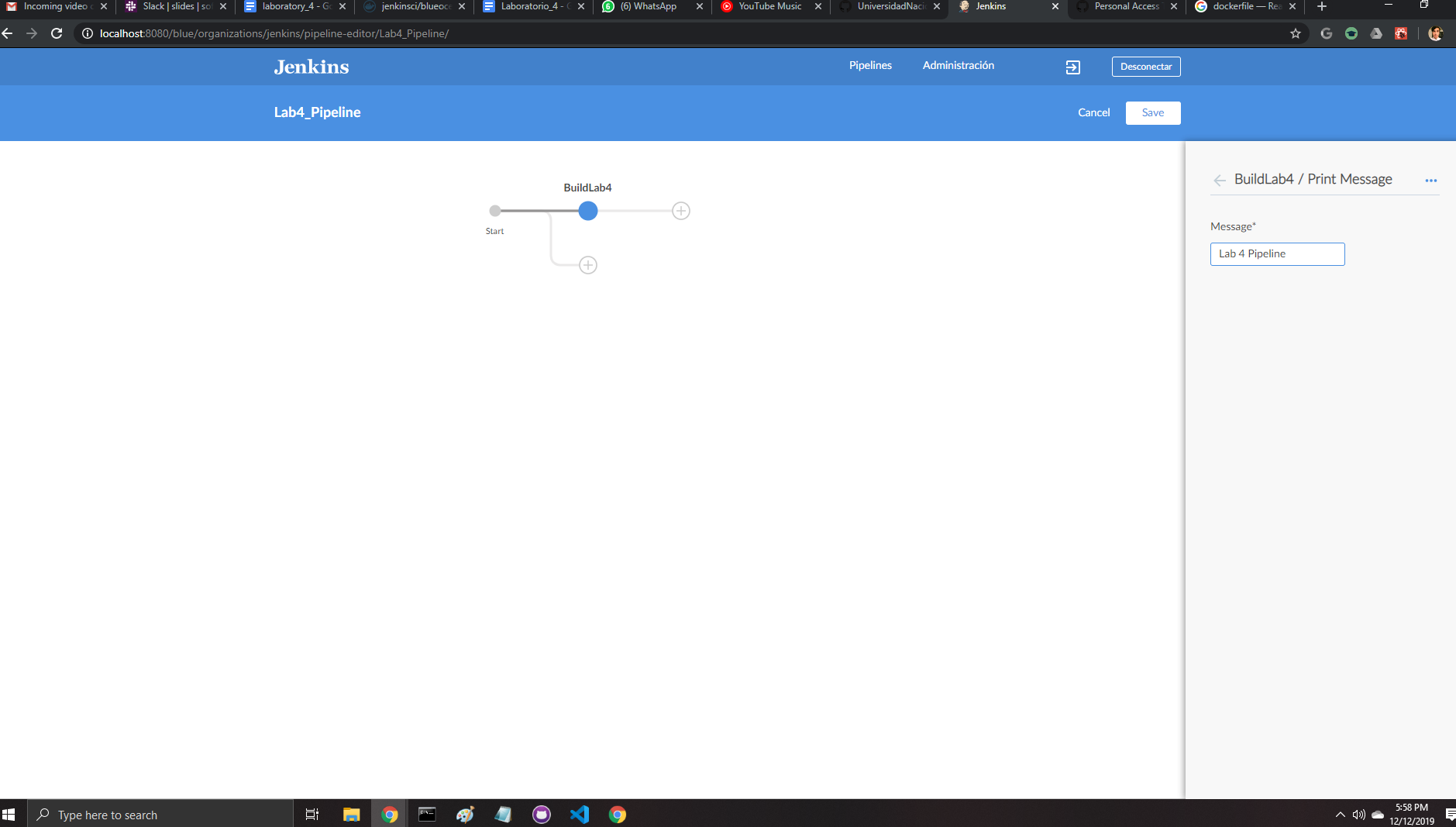


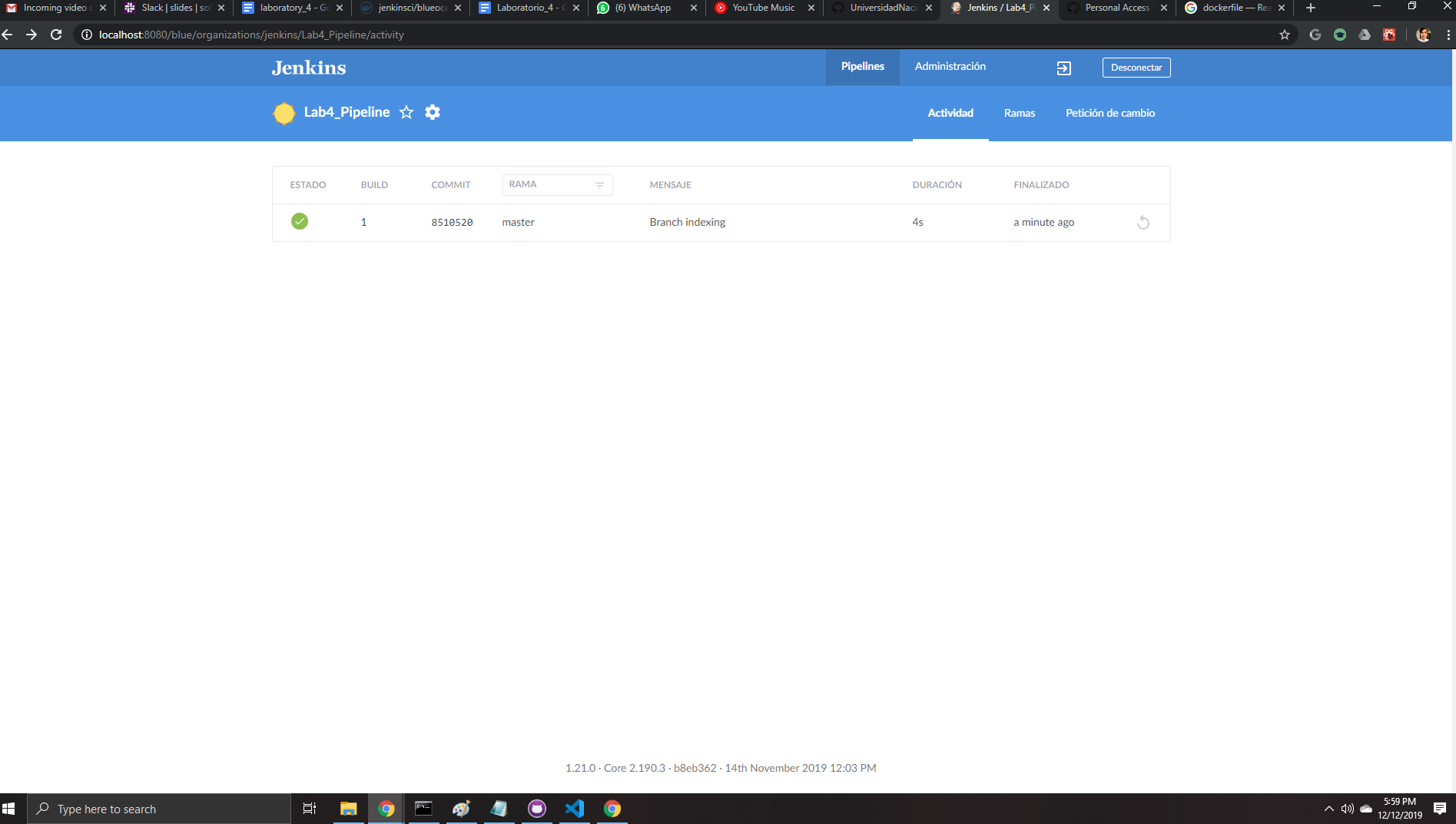
**2 - Create a pipeline like this. Use a dummy software application to attach to the pipeline. You can choose the application that you prefer. Be creative choosing stages and steps for doing static code analysis, testing, security (such as run audit for a front project), etc.**

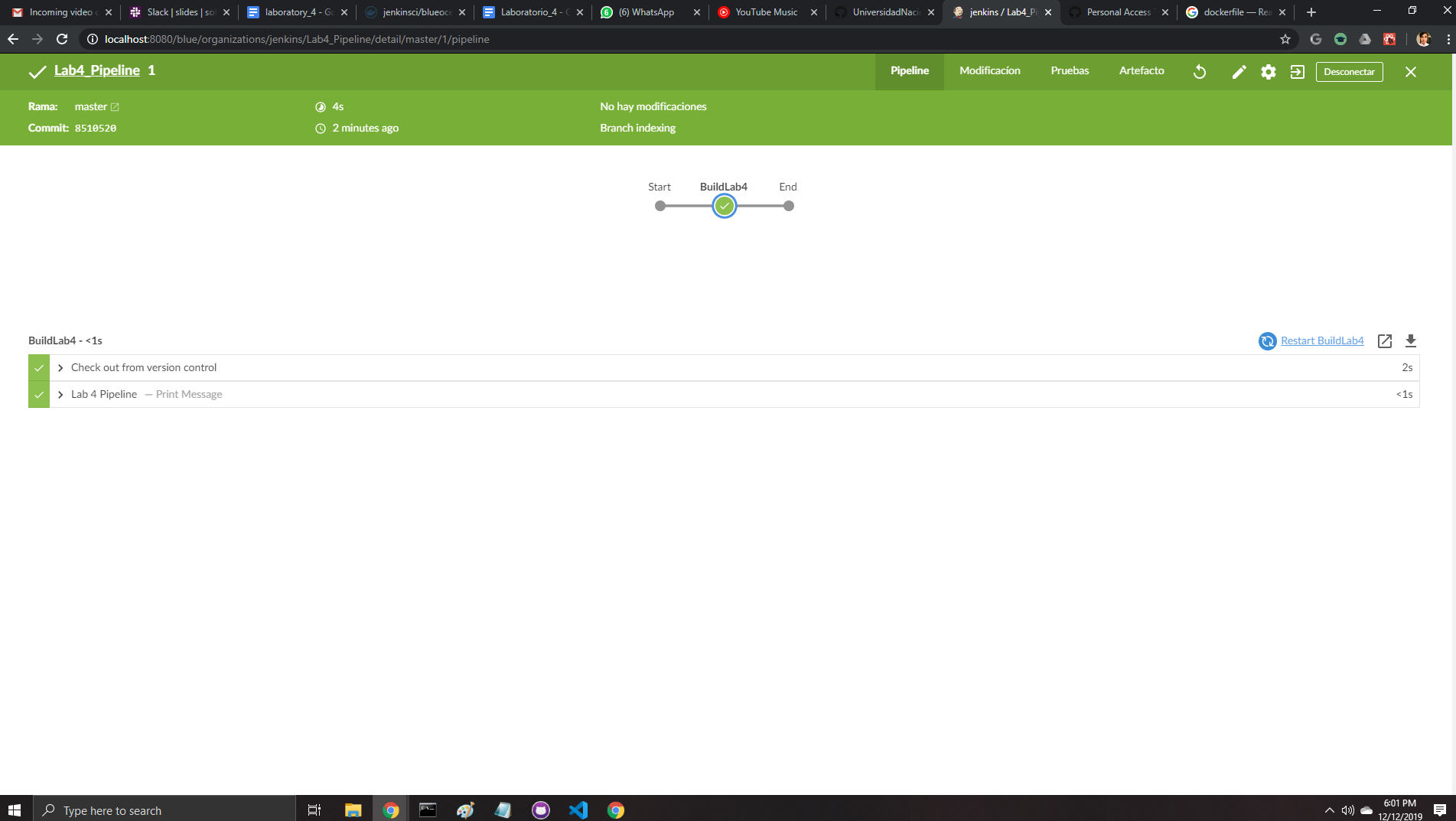


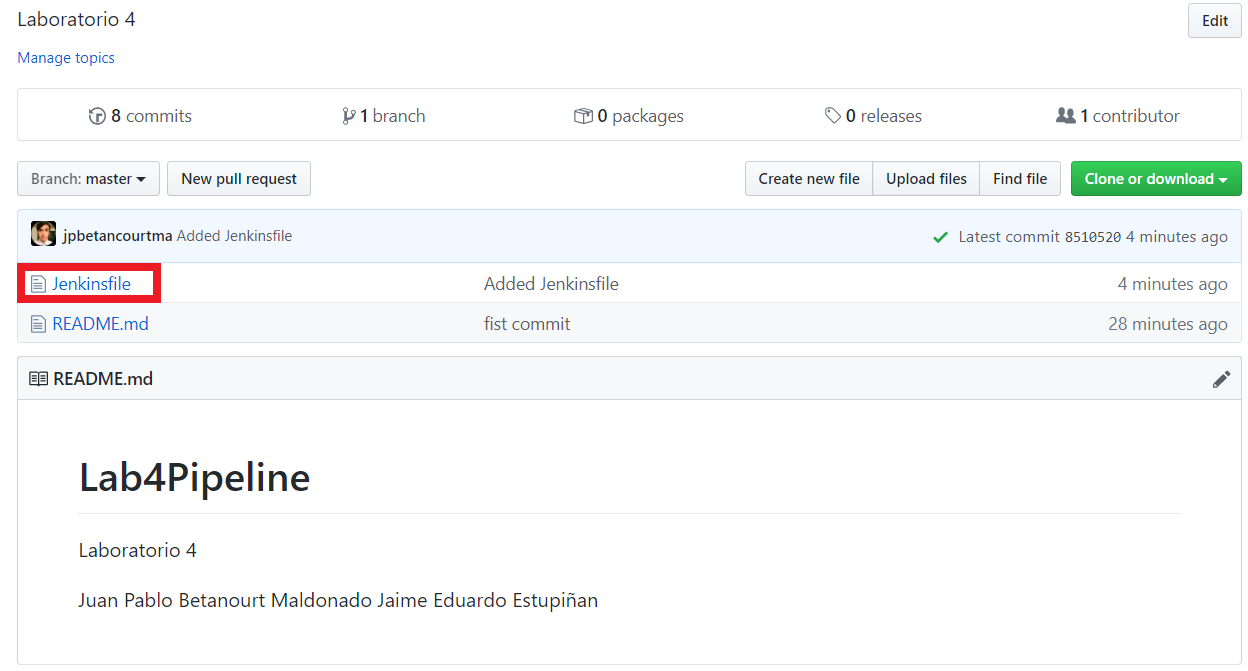
****

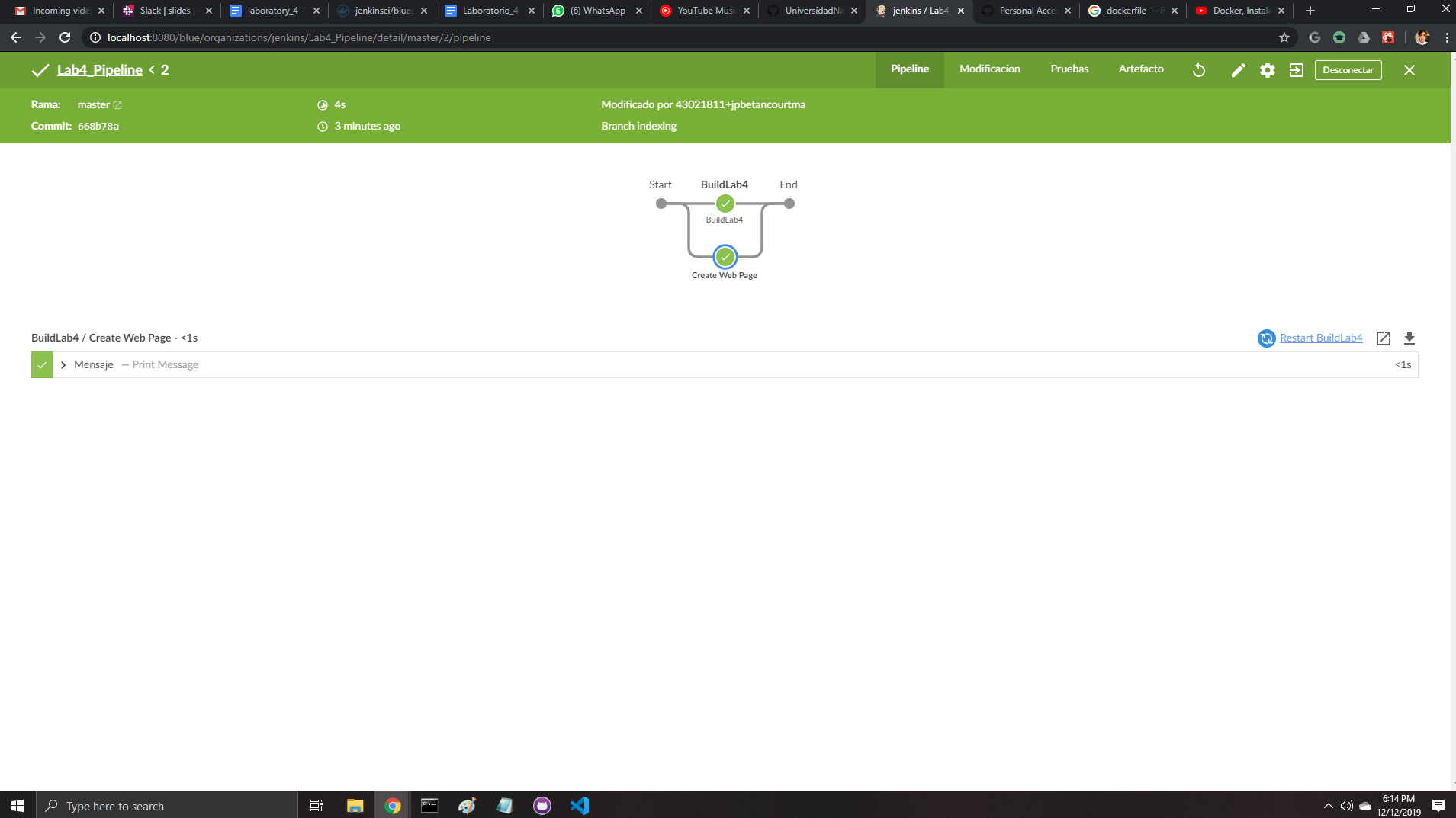
****

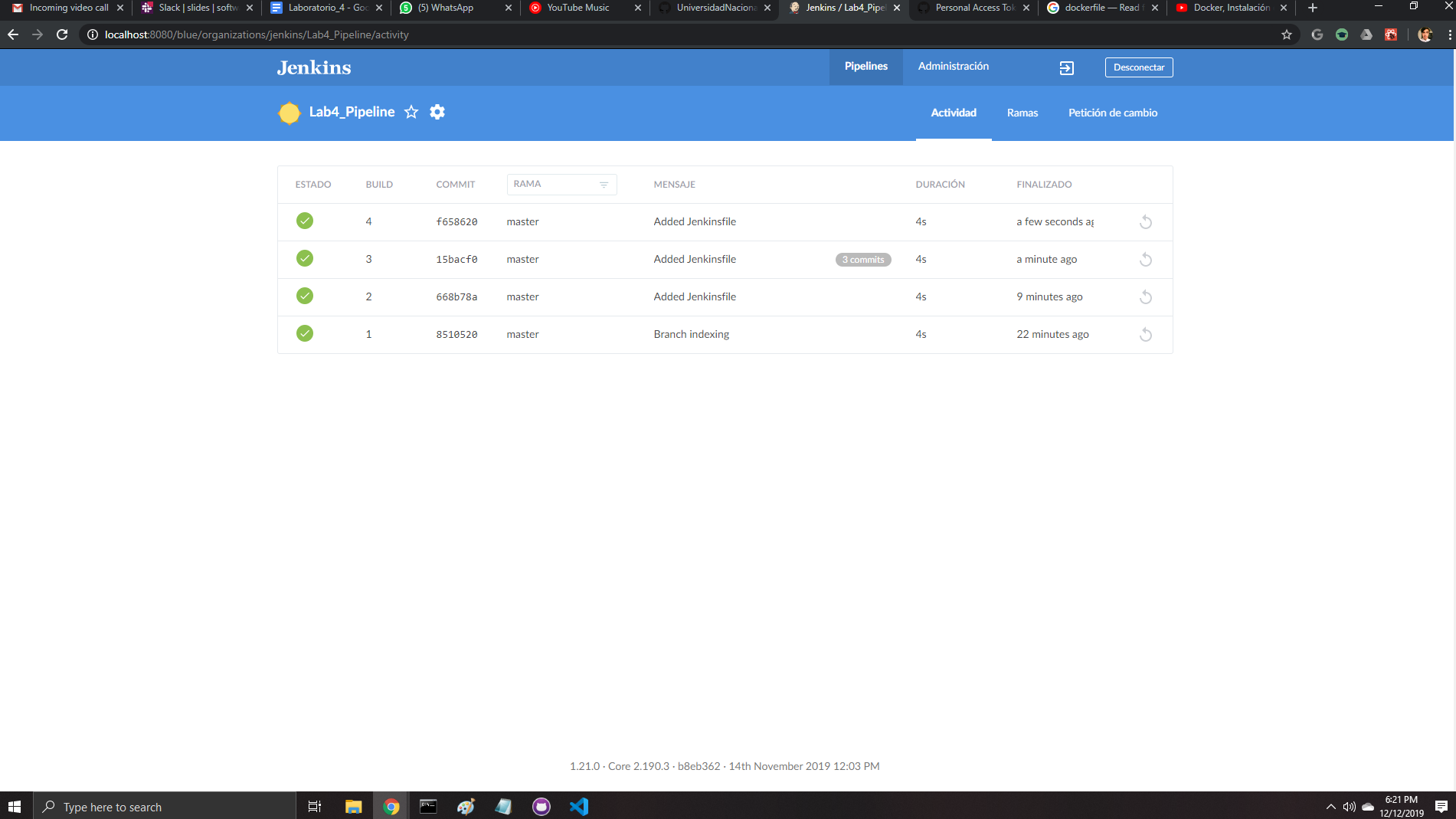
****

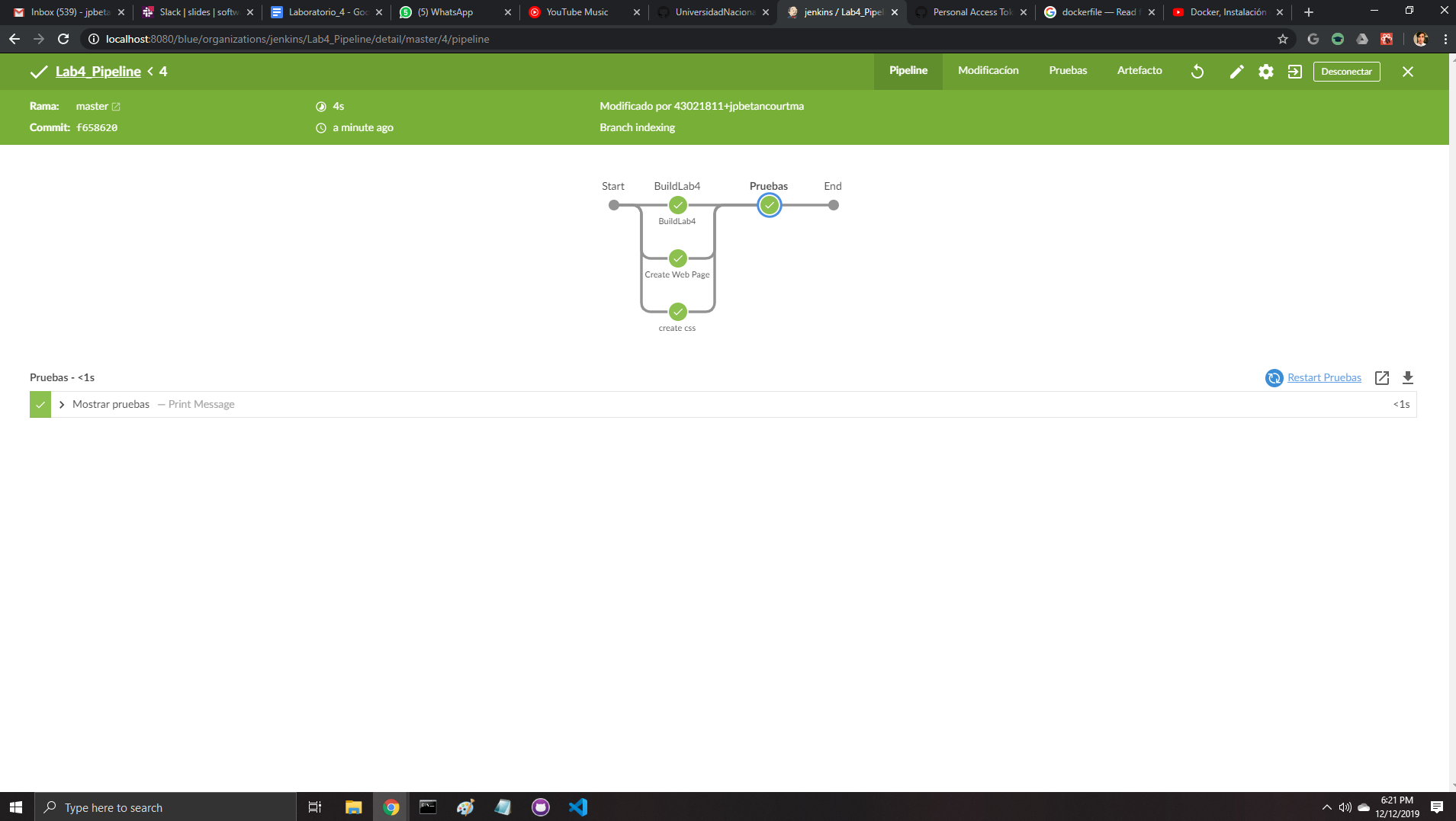
****

****

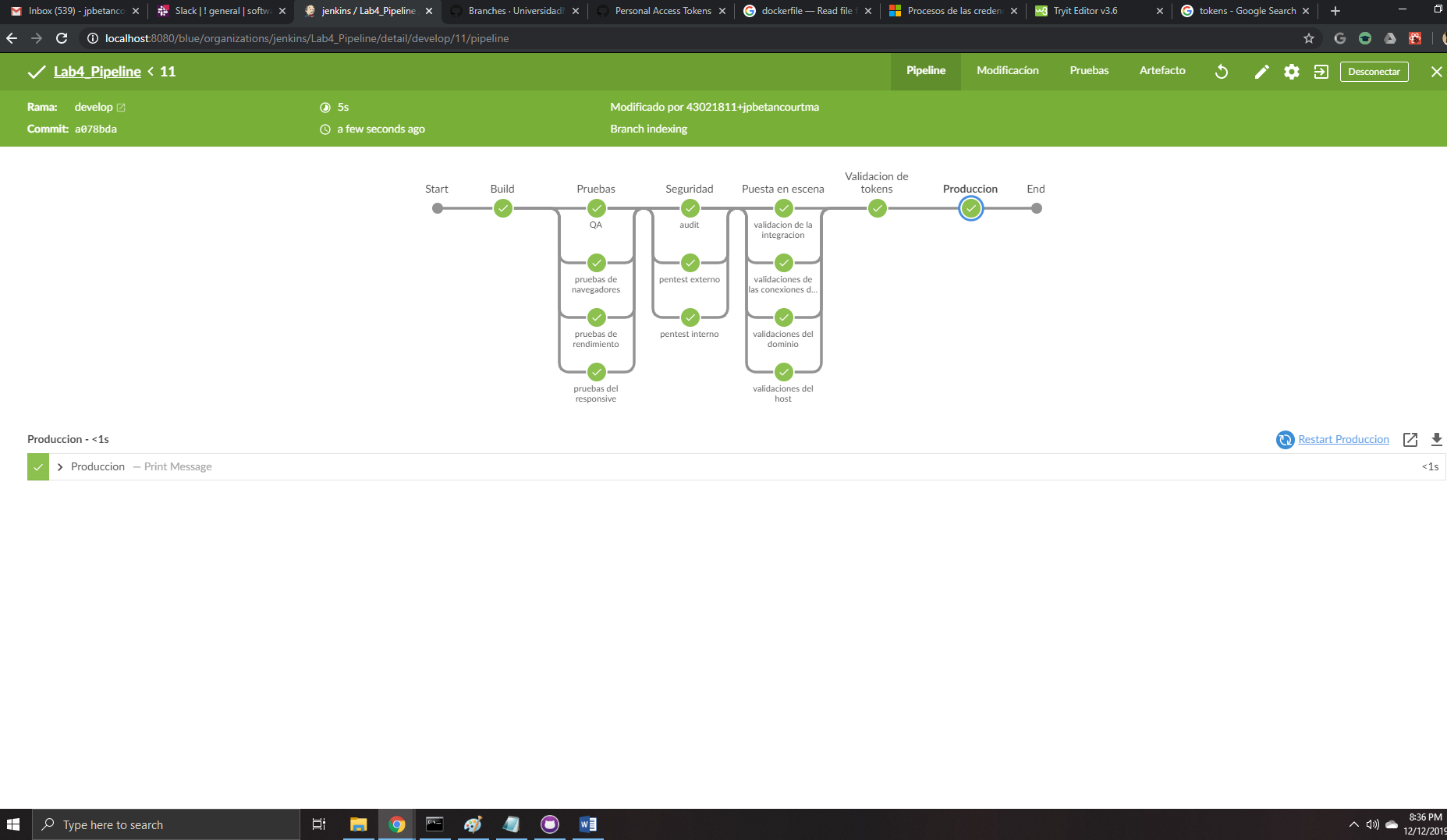
****

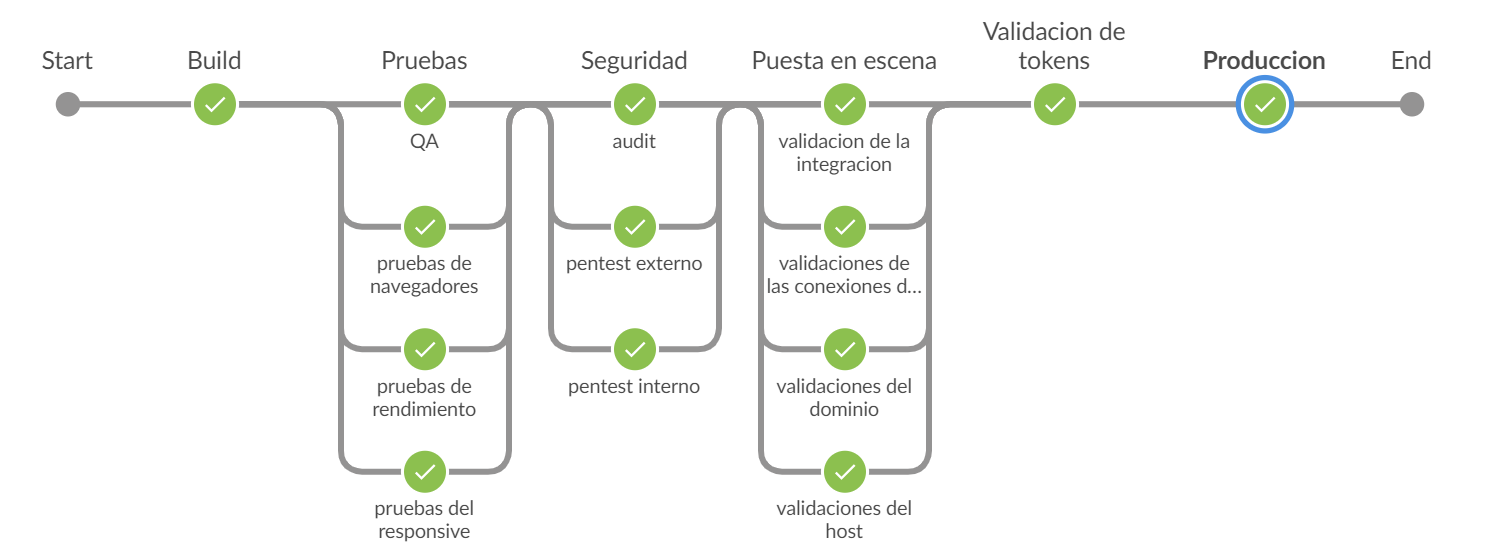
****

****

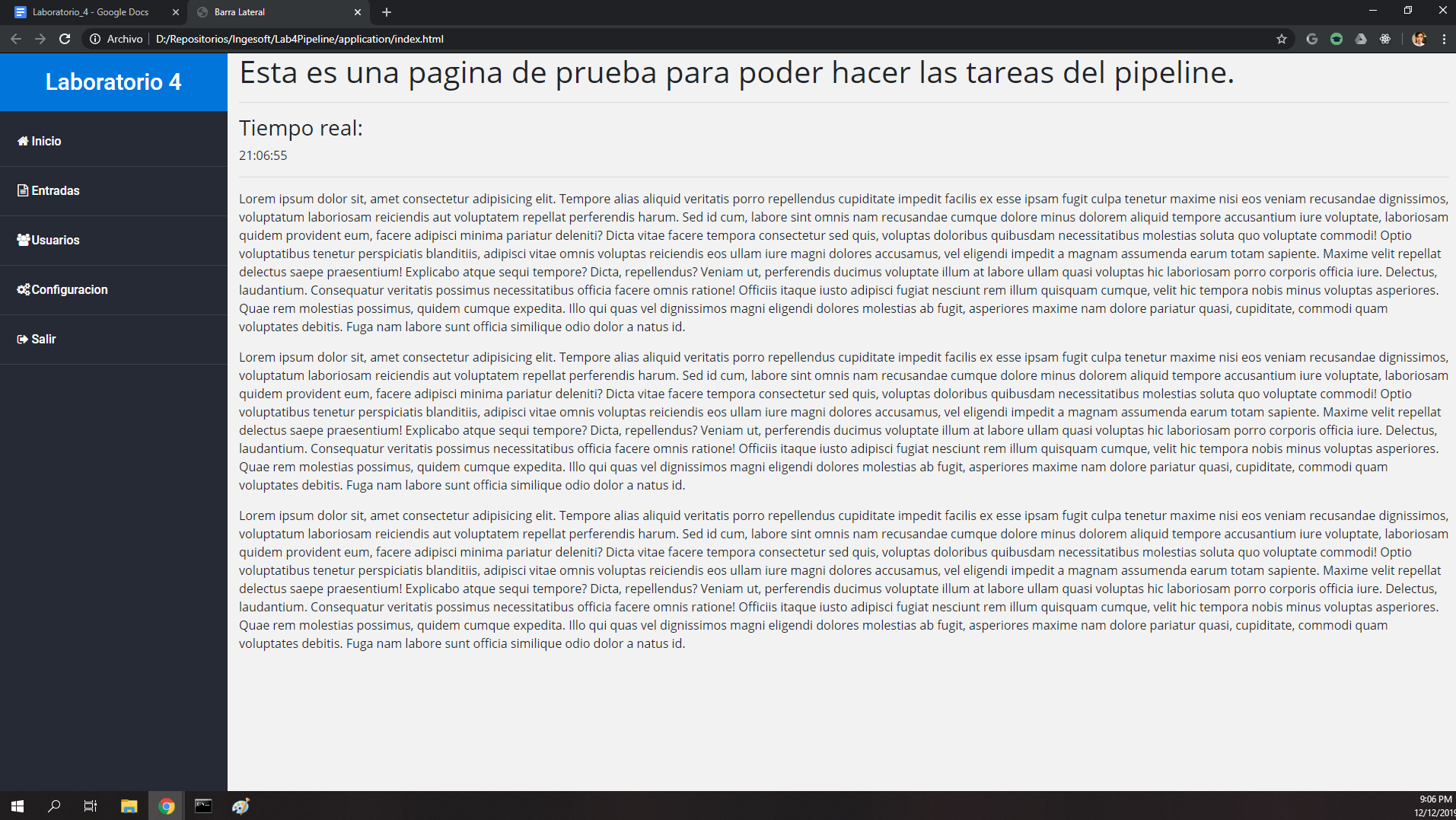
****

****

****

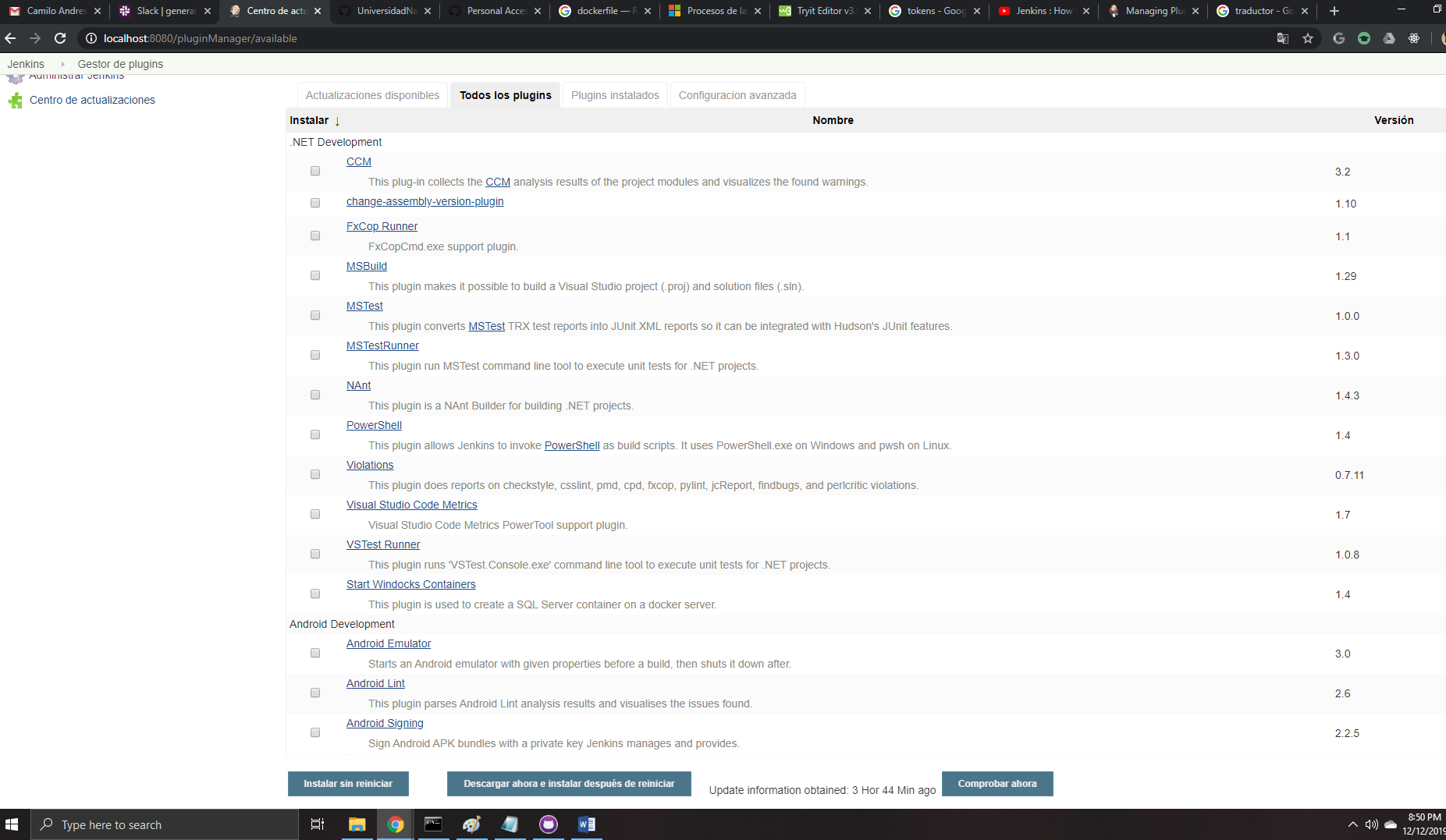
****

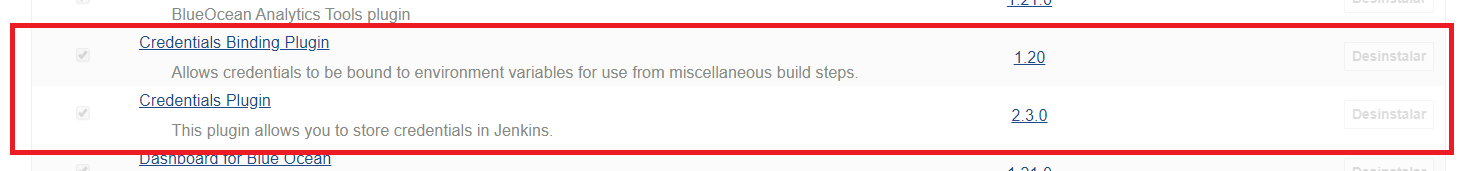
Aplicación de prueba:

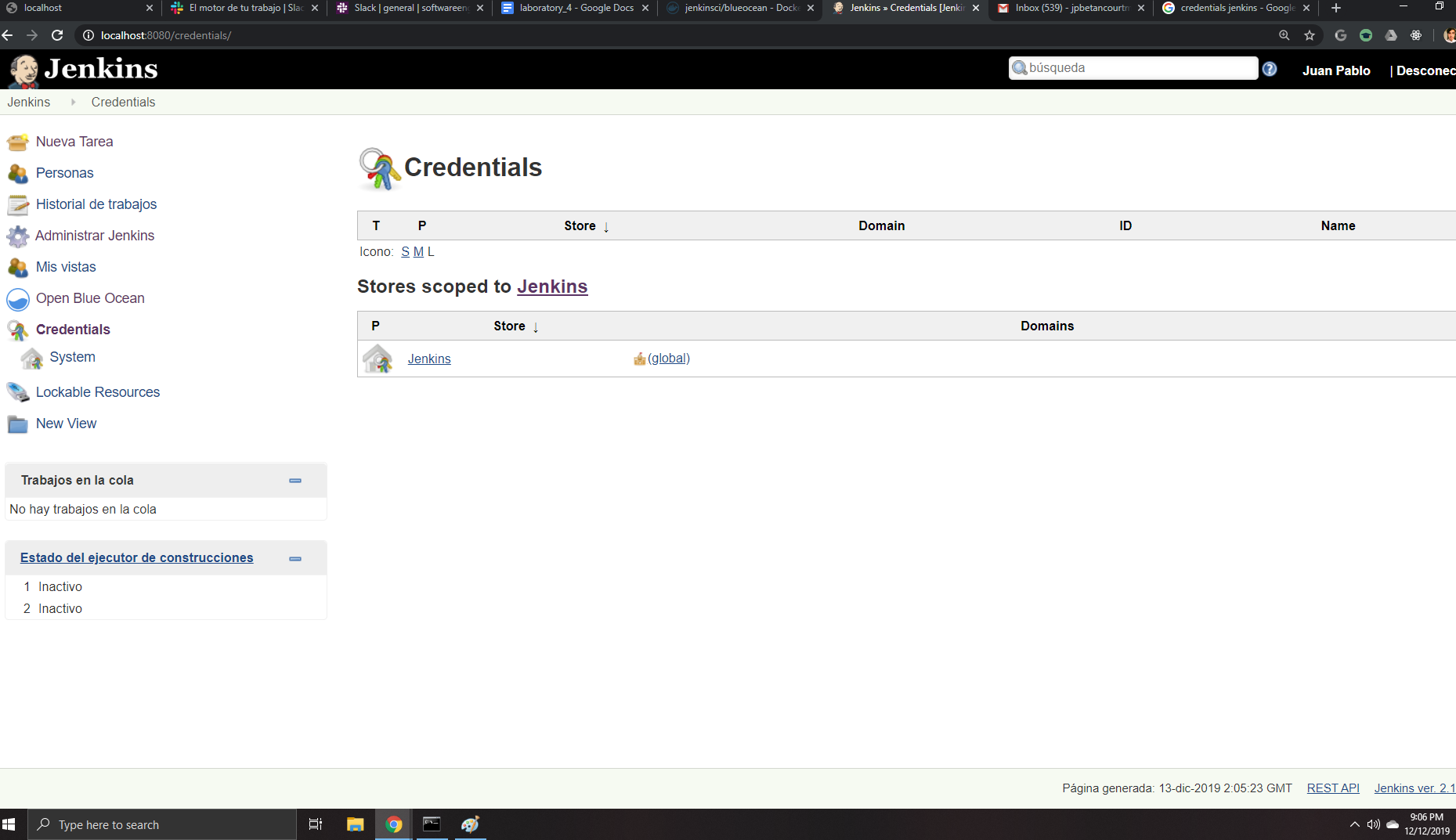
****

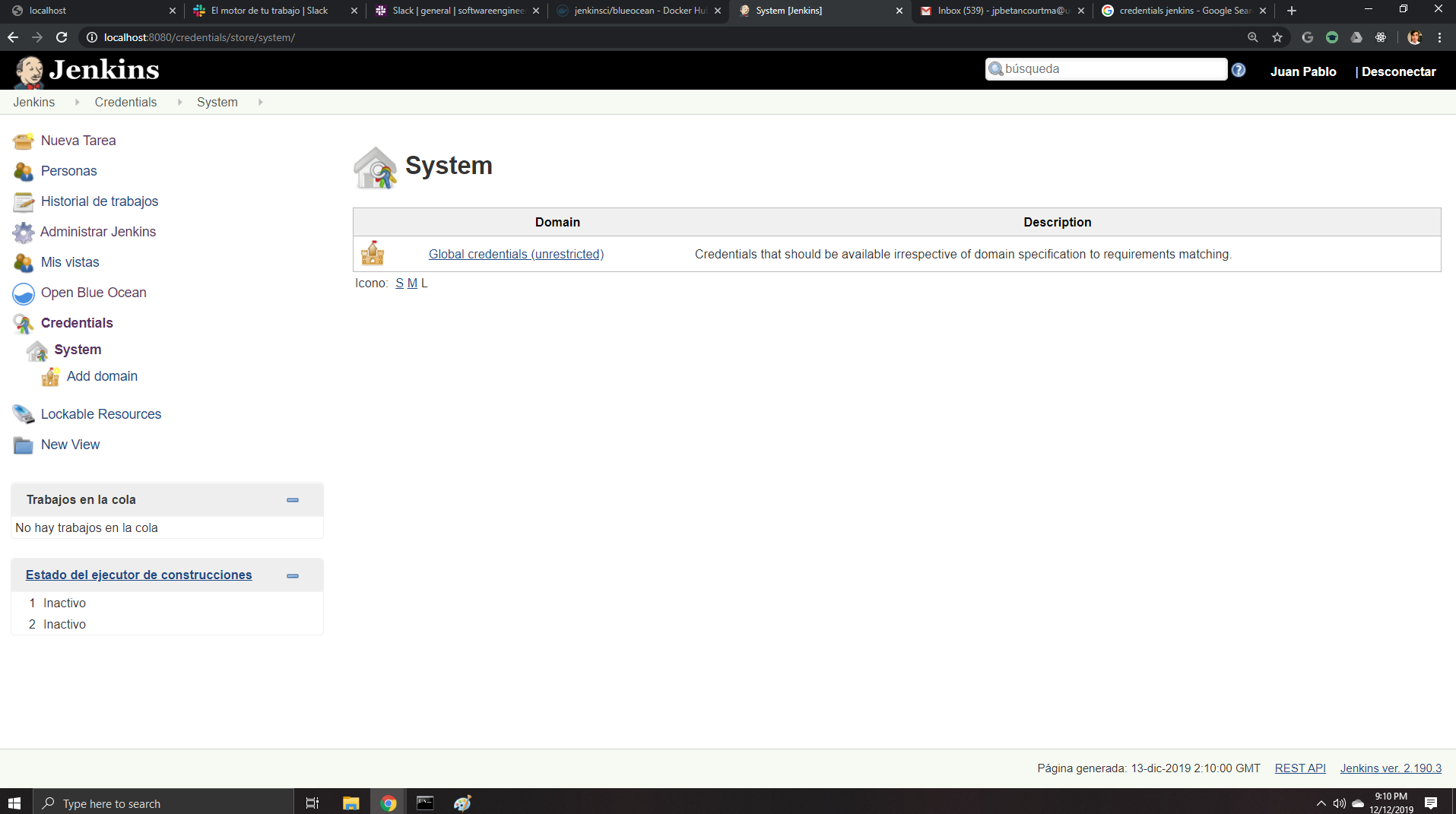
**3 - Install other plugin (you can choose anyone) and send evidence of the usage.**

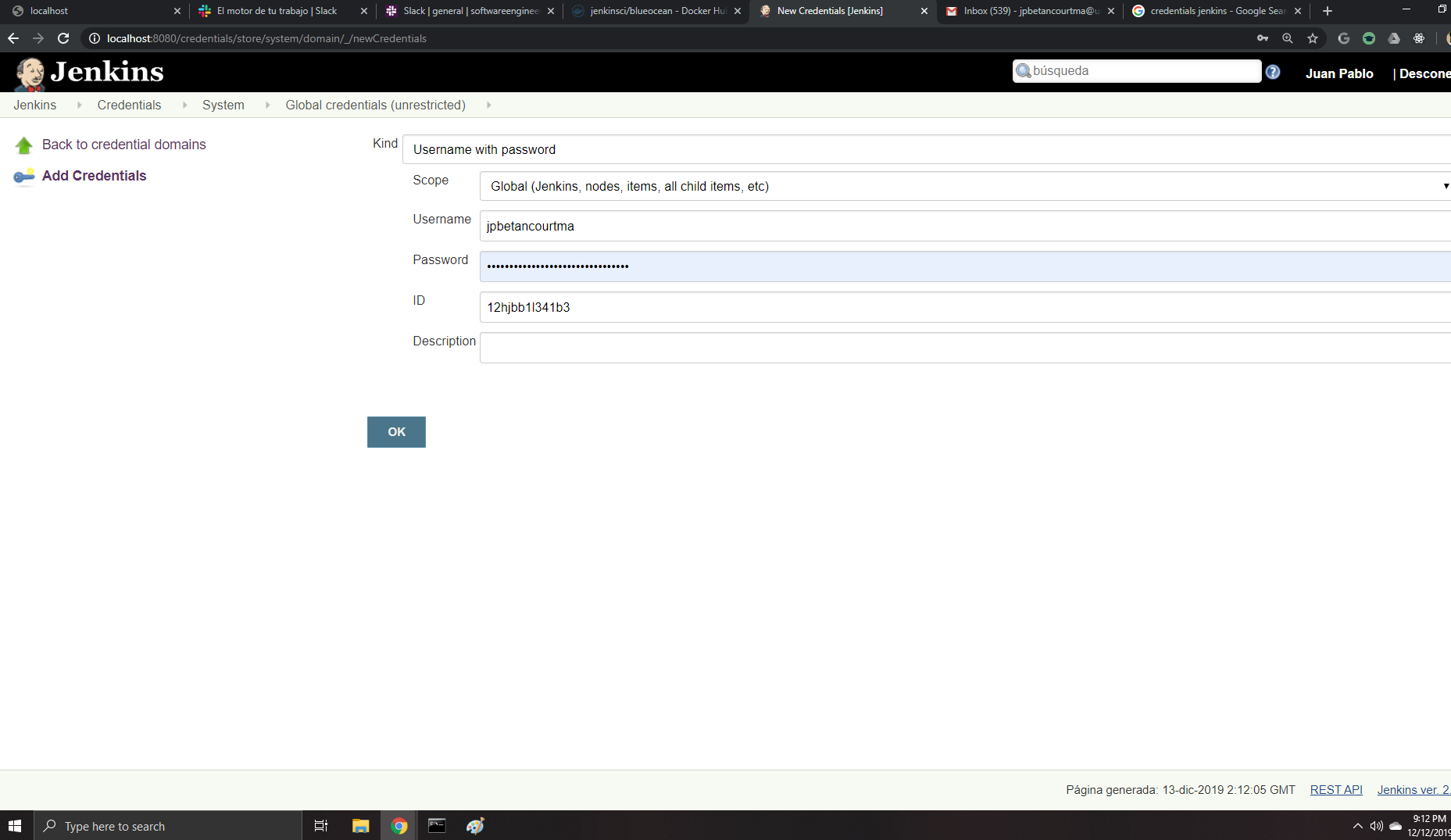
Nosotros instalamos el plugin de credenciales de Jenkins:

****

****

****

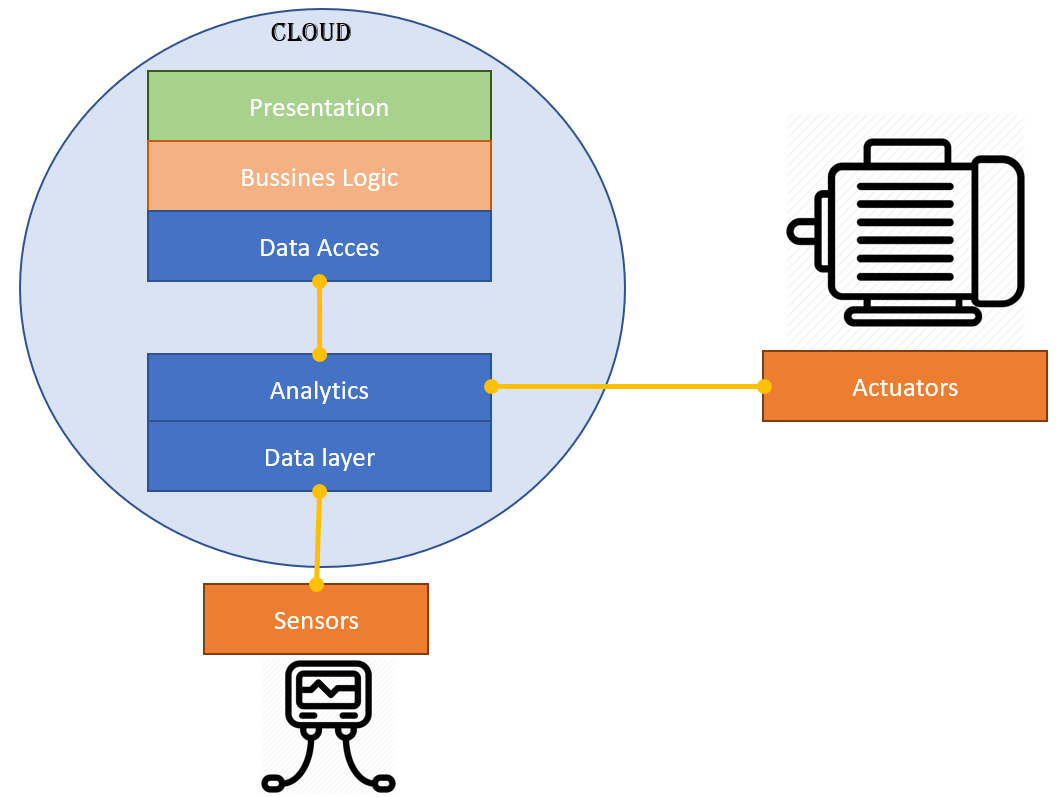
****

****

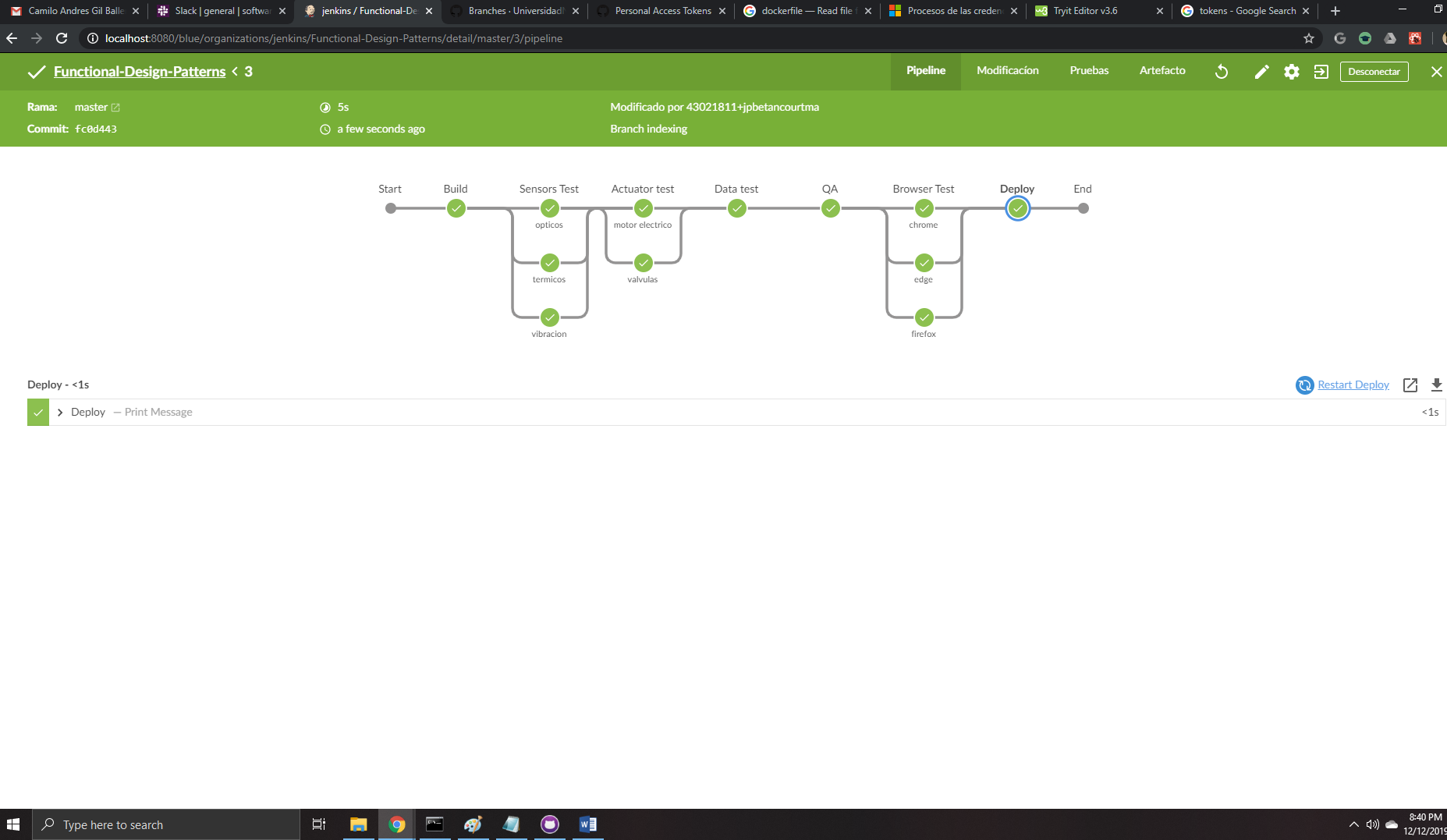
**4 - Read this document** [**https://docs.gitlab.com/ee/topics/autodevops/**](https://docs.gitlab.com/ee/topics/autodevops/) **and mention the differences, pros and contras between to implement a pipeline with Jenkins and to use a solution for auto DevOps.**

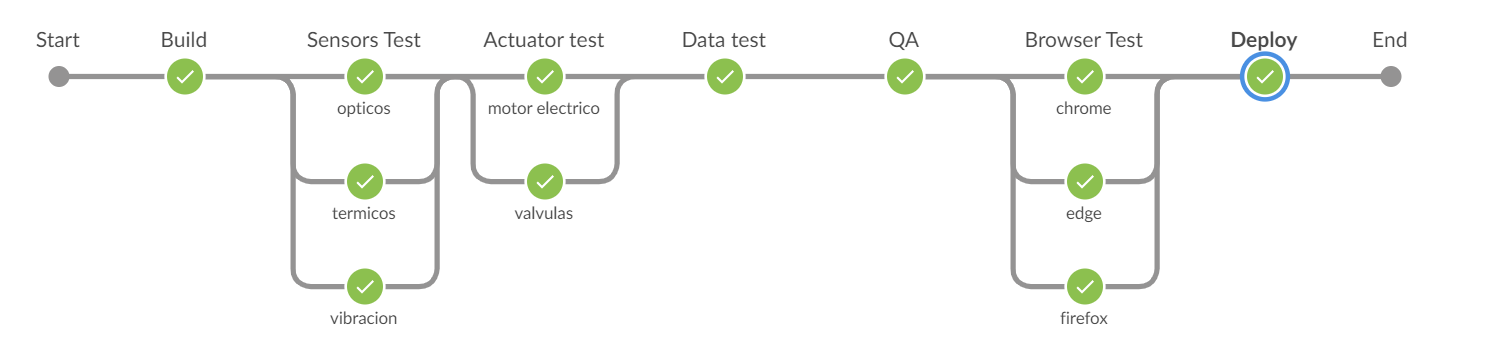
|  |  |  |
| --- | --- | --- |
|  | **Pros** | **Contras** |
| **Pipeline with Jenkins** | * Código abierto, es bastante intuitivo con el usuario y no necesita descargar como tal otros componentes. (Los plugins se consideran parte del paquete) * Fácil configuración, es decir que puede ser modificada fácilmente y si es necesario se puede expander. * Cuando se hace un Deploy genera reportes de pruebas, y puede ser configurado teniendo en cuenta los requerimientos de CI y CD. * No importa el sistema operativo. * Existen un montón de plugins que permiten convertir a Jenkins en una herramienta flexible, permitiendo el deploy y automatización a través de varias plataformas. * La comunidad para colaborar es bastante extensa, ya que es código libre varios equipos de desarrollo agile colaboran en responder las preguntas. * La mayoría del trabajo de integración es automático. Esto significa que hay menos errores en la integración , ahorrando tiempo a los desarrolladores. * Docker en conjunto de Jenkins son una gran herramienta implementada por los equipos de desarrollo ágiles. El ecosistema que existe con la integración de estos provee una coordinación en la infraestructura de software muy buena. | * El precio de usar el servidor en el que Jenkins se ejecuta no es fácil de predecir. (según varias reviews, esto no lo podemos comprobar ya que no se ha usado extensamente la herramienta). * El manejo de Jenkins usualmente se hace por una persona, es decir que esa persona tiene que ser responsable en cuanto problemas al hacer push del código. Existe un tracking para el código que ingresa a Jenkins, pero no uno para Jenkins como tal, lo cual resulta en problemas para comprender la infraestructura que maneja la persona asociada a Jenkins. * Jenkins no permite ver los commits de los demás colaboradores, que en general puede generar problemas para hacer tracking del software. * Jenkins no provee ningun tipo de analiticas, existen algunos plugins pero seguún las reviews no son suficientes. Esto también contribuye al problema de no tener tracking en general. * La necesidad de un personal que esté atento, tanto al servidor de jenkins como a Jenkins, es vital. Esto porque es una herramienta muy poderosa que necesita estar siendo monitoreada constantemente. Esto añade horas de trabajo al personal y termina en costos. |
| **Auto DevOps** | * Los DevOps basados en la nube facilitan la colaboración ya que cualquier usuario puede acceder. * El Auto DevOps hace que el testing y deploy seha mas rapido. El uso de tecnologías en la nube permite que se incremente la frecuencia de los lanzamientos (release). * Las empresas proveen un equipo de desarrolladores especializados que siempre están atentos a resolver los problemas que se vayan creando y colaborar al equipo de desarrollo propio. * Muchas veces la complejidad de la información y el flujo de la información no se muestra en un Auto DevOps. Esto ayuda a que el equipo se concentre en desarrollar herramientas específicas en lugar de tener que entender toda la cadena de trabajo. * El equipo de desarrollo puede hacer pruebas usando el código fuente de las herramientas de gestión y el equipo de TI puede hacer cambios en la configuración de las herramientas de manejo. * Todos usan el mismo set de información, facilitando la documentación. * En el caso de hacer cambios rápidos o imprevistos, es más fácil por medio de los Auto DevOps ya que estos tienen la capacidad de adaptarse automáticamente a lo que el equipo necesite. | * La seguridad se puede ver afectada un poco, ya que se deja a un tercero el manejo de las operaciones de DevOps. No obstante al trabajar con una herramienta de una empresa reconocida, pueda que esto no sea un problema, gracias a las garantías que se ofrecen. * Al dejar en manos de un tercero el CI y CD, tenemos que el equipo debe tener una comprensión sobre la integración de varios módulos, los sistemas de cloud y las soluciones DevOps ofrecidas por la empresa contratada. De lo contrario, no existe un soporte del sistema cuando se esté en producción. * Para poder exceder las limitaciones que da el proveedor del Auto DevOps es necesario tener un conocimiento de experticia extra en cuanto a infraestructura, incluyendo entendimiento de la integración y workflow que maneje la herramienta como la empresa. * La aplicación se vuelve completamente dependiente del proveedor del servicio, ya que no se tiene un control completo sobre los pasos del ciclo DevOps. |

**5 - According to the presentation “Industria 4.0 Internet de las Cosas Industrial” explained by Professor Tatiana in the last class, provide an architecture diagram with one of the uses cases explained by her and implement a pipeline that involves the DevOps stages that you consider (be creative in this part).**



El diagrama de arquitectura se realizó pensando en el caso que comento la profesora Tatiana en el que se tiene un motor, el cual puede producir diferentes tipos de vibraciones, mediante la ayuda de un sensor se pueden calcular la magnitud y cantidad de estas vibraciones para determinar si el dispositivo necesita de un mantenimiento, en caso de necesitarlo se podría ejecutar un actuador que emitirá algún tipo de señal física indicando que el dispositivo necesita un mantenimiento, o podría tomarse los resultados de este análisis y mostrarse a los usuarios otro tipo de dispositivo, por ejemplo un móvil o en una página web para que mediante este una junta directiva tome diferentes tipos de decisiones (Como mantenimientos de tipo preventivos/predictivos).





*Link repositorio de github:*<https://github.com/UniversidadNacionalDeColombia/Lab4Pipeline>