Developing and integrating an application: DevOps approach

Pablo Orviz
orviz@ifca.unican.es

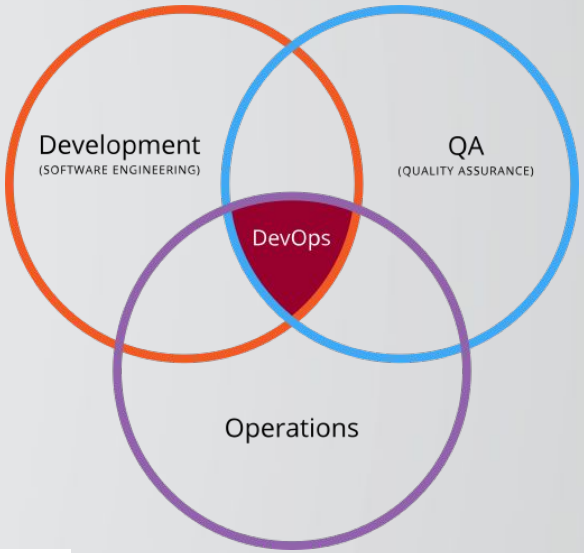
IFCA-CSIC

INDIGO-DataCloud is co-founded by the Horizon 2020Framework Programme

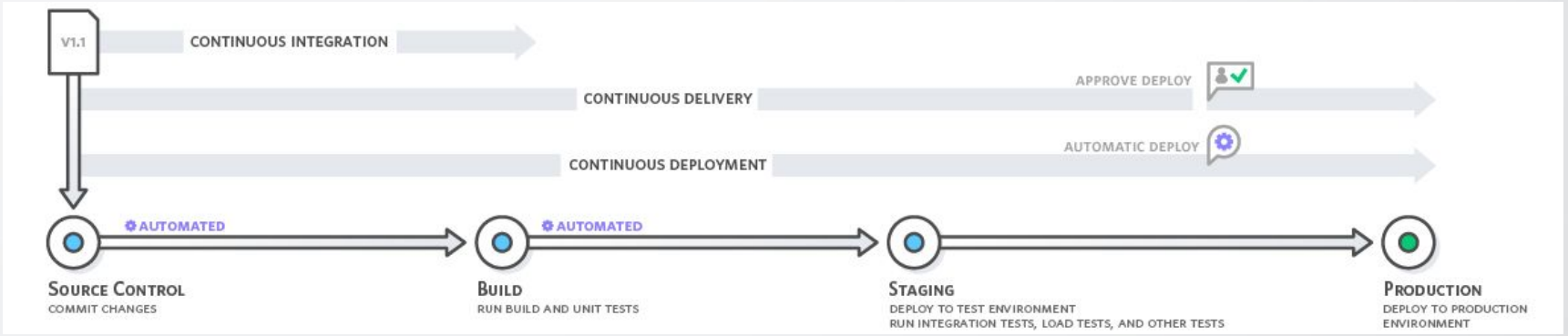
DevOps culture

INDIGO - DataCloud

"set of practices that emphasize the collaboration and communication of both SW developers and IT professionals while automating the process of SW delivery and infrastructure changes" (source: wikipedia)



- **Dev**elopment and **Op**eration**s**
  - traditionally distant → new features vs stability
  - QA procedures → maturity of SW → trust/confidence
  - fast and frequent delivery → SW more reliable
- Approaches (→ automation)
  - Continuous Integration (CI)
  - **Continuous Delivery (CD)** *suitable for user community apps*
  - **Continuous Deployment (CDep)** *suitable for user community apps*

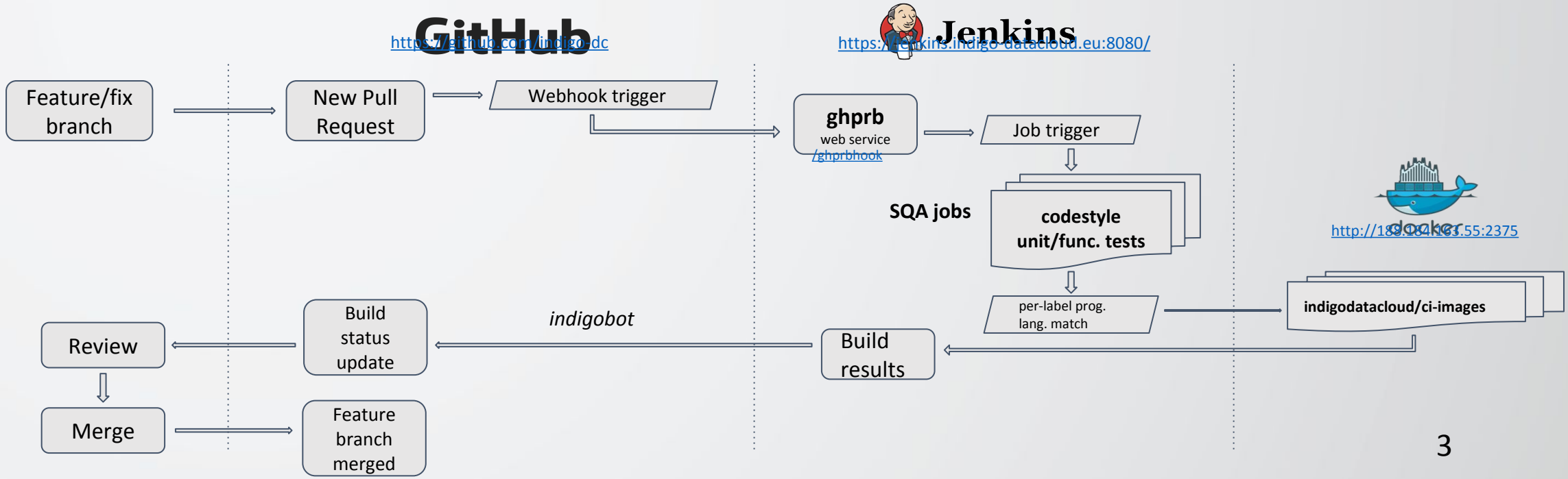


DevOps in INDIGO: core services development

1. Core services development ⟹ follows a CI (Continuous Integration) approach

Code at the production branch is proven to be production-ready

i. Each new change in the code (feature, bugfix) is automatically tested
ii. Successful tests are required before merging the code in the production branch
iii. Human code reviews done as the last step in the chain

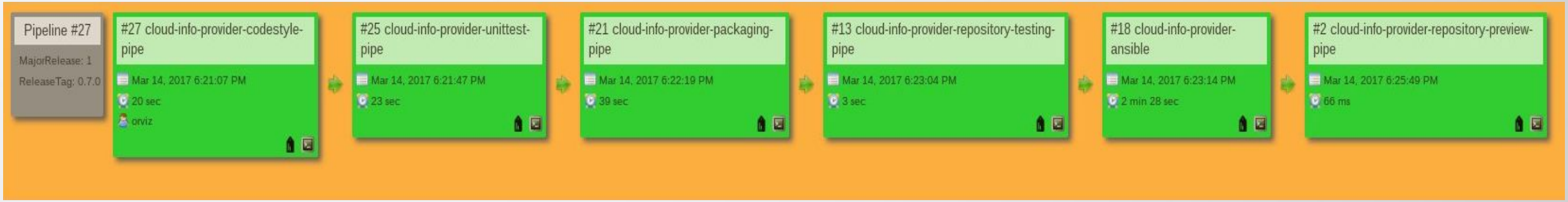


DevOps in INDIGO: core services release

2. Core services validation and release ⟹ conforms a CD (Continuous Delivery) scenario (pipeline)

Software (packages, appliances) can be reliably released at any time

i. Source code checks are re-ran (regression testing) and built
ii. When releasing packages:
    1. RPMs/DEBs are created and uploaded to the *testing* repository
    2. Deployment is tested using available Ansible roles or Puppet modules
    3. RPMs/DEBs are moved to the *preview* repository - through a promotion/approval process -
iii. When releasing containers (in progress)
    1. Docker image is built and uploaded to *indigodatacloud* DockerHub organization



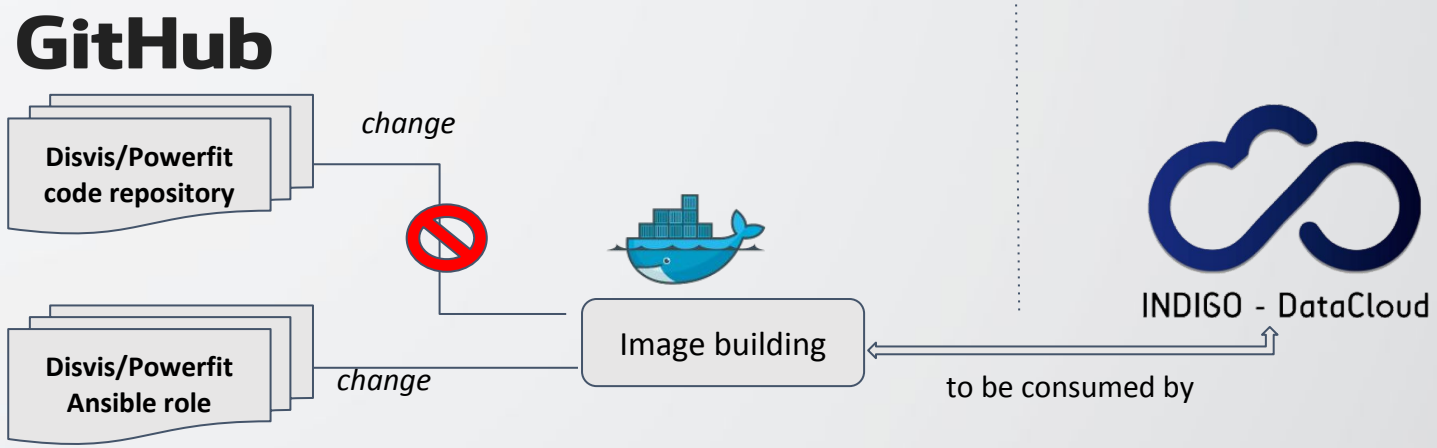
DevOps in INDIGO: user community apps

3. Move a step forward and apply INDIGO insights on DevOps to the user community workflows
   - 2 approaches
     - Continuous Delivery (CD)
     - Continuous Deployment (CDep)
   - INDIGO prerequisites (INDIGO expert)
     - A *TOSCA template* to describe the application composition
     - A Docker image to run the application, automatically deployed using *Ansible roles*
   - Actors involved
     - Source code repository (**GitHub *indigo-dc***): hosts application code
     - Continuous Integration Srv (**INDIGO Jenkins CI**): code testing and Docker image building
     - Container registry/catalogue (**DockerHub's *indigodatacloudapps***)
     - **INDIGO platform**: deploys and runs the application

Continuous Delivery (CD) on user community apps

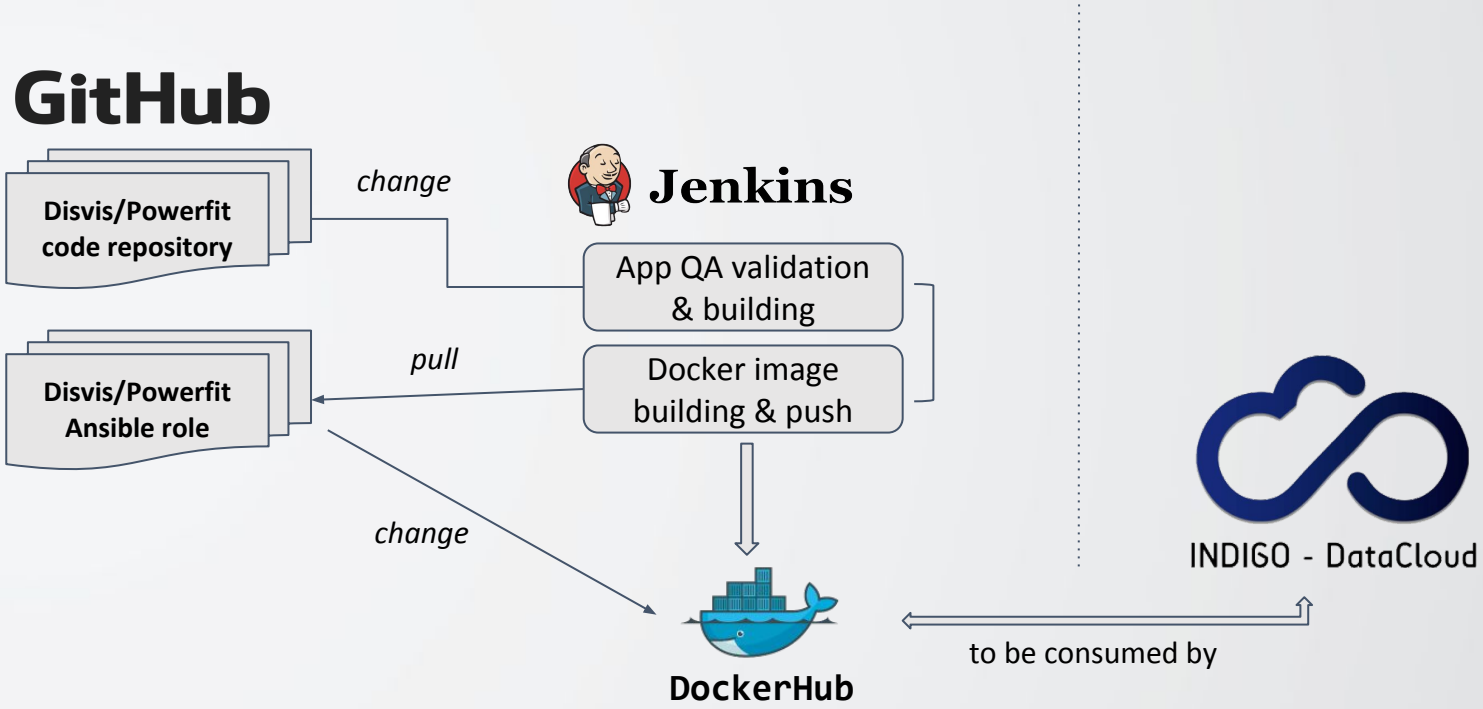
Target: ***Deliver fast, good-performing appliances***

- INDIGO-DataCloud uses mainly Docker containers for user applications
  - Applications are deployed using **Ansible roles**, pushed to *indigodatacloudapps* DockerHub's organization
  - DockerHub <-> GitHub integration via *automated builds*
    - Images are automatically built when a change is done to the Dockerf whenever a change is done in the application code repository
    - QA checks: the Ansible deployment itself



Towards a DevOps-like Continuous Delivery approach

Adding a CI service..



- Docker image is built when a change has been made to the application code repository
- Application is QA-validated DevOps compliant approach

Continuous Deployment (CDep) in user community apps

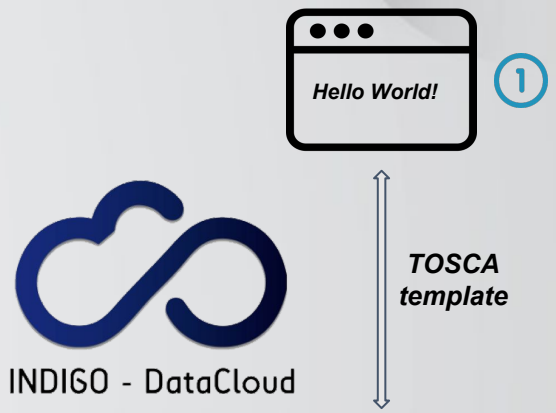
INDIGO - DataCloud

Target: ***Fully automate the application deployment in production resources for any given change***

- One step beyond: not only delivering QA-validated Docker images but deploy them on production resources
  - Challenge: live deployment

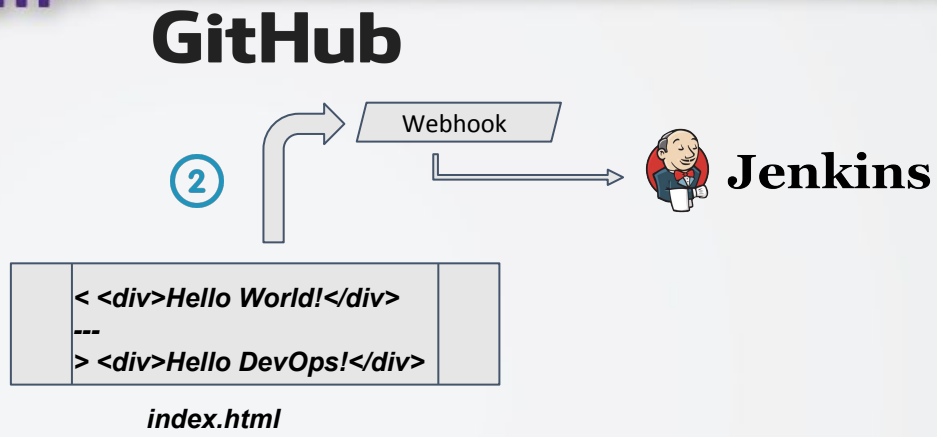
- No current use case
  - Especially suitable for long-running services (portals)

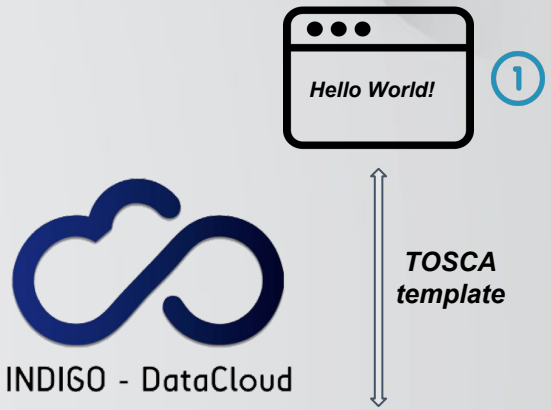
Proposal of INDIGO Continuous Deployment pipeline



1. Web application running on a INDIGO-reachable resource provider

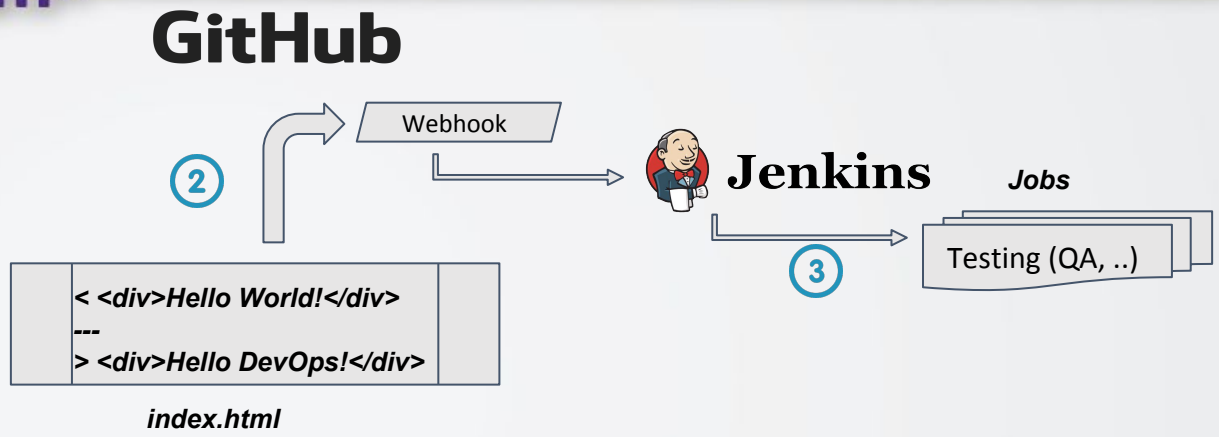
Proposal of INDIGO Continuous Deployment pipeline

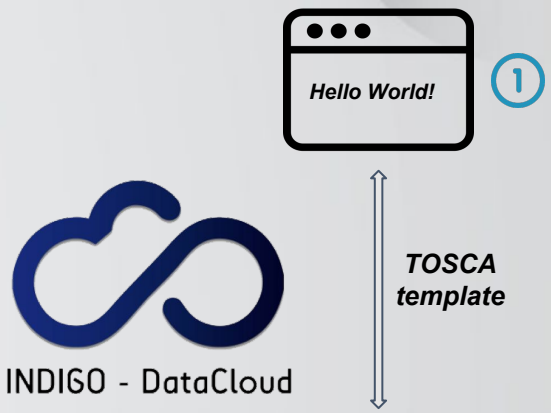




1. Web application running on a INDIGO-reachable resource provider
2. A change in the codebase is committed in GitHub, which triggers job execution in Jenkins [user]

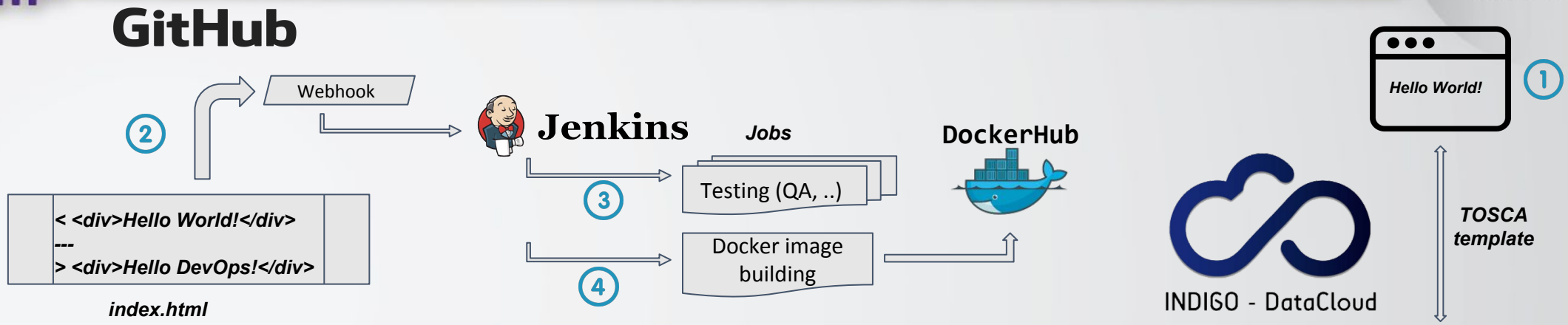
Proposal of INDIGO Continuous Deployment pipeline





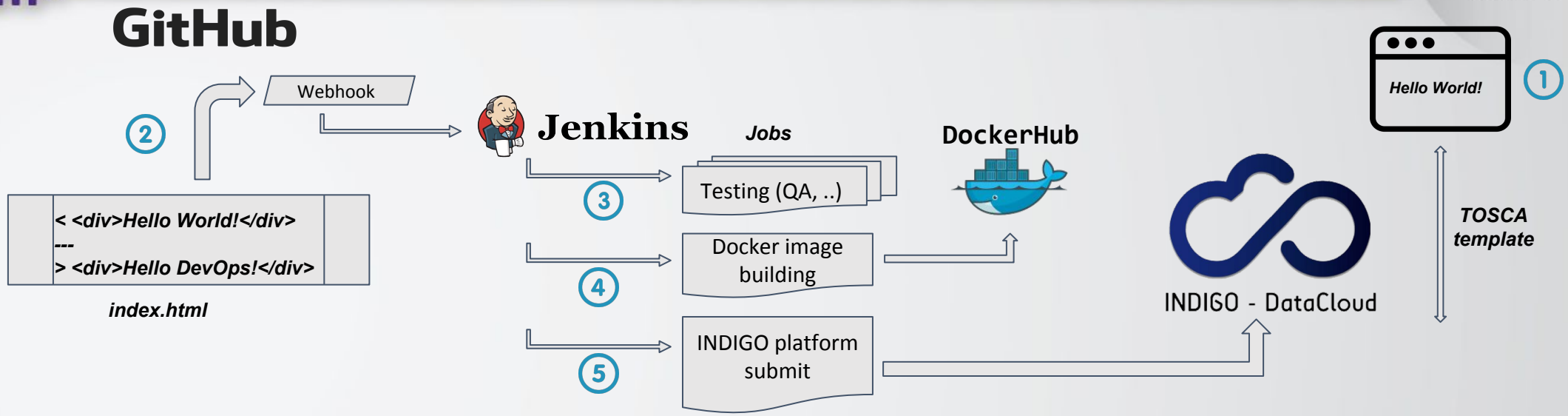
1. Web application running on a INDIGO-reachable resource provider
2. A change in the codebase is committed in GitHub, which triggers job execution in Jenkins [user]
3. 1st job in the stack: tests/validates the source code, build the application, etc.

Proposal of INDIGO Continuous Deployment pipeline



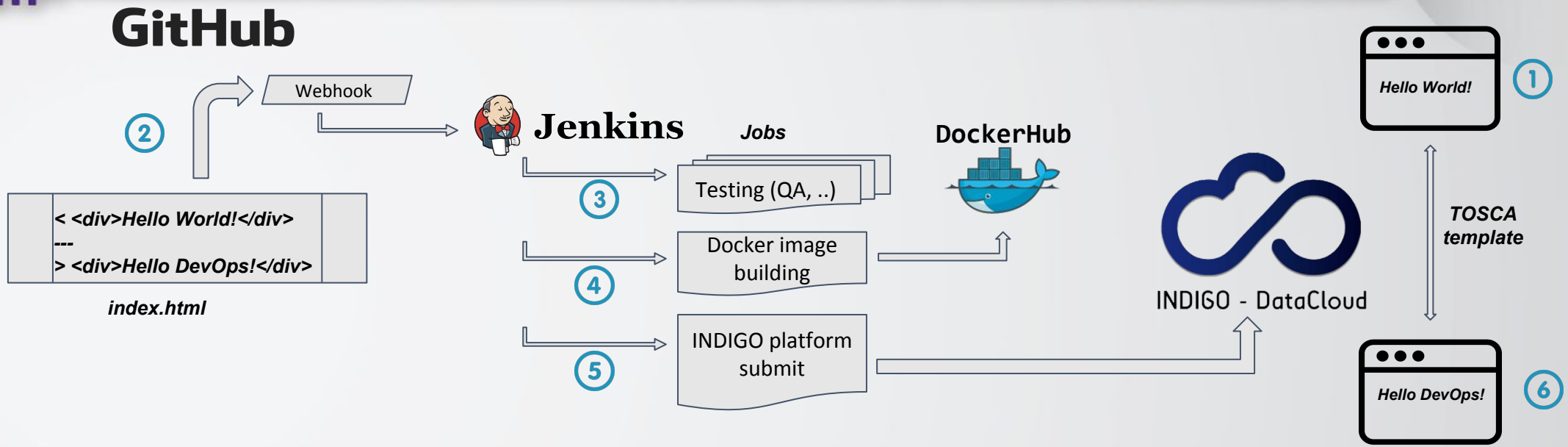
1. Web application running on a INDIGO-reachable resource provider
2. A change in the codebase is committed in GitHub, which triggers job execution in Jenkins [user]
3. 1st job in the stack: tests/validates the source code, build the application, etc.
4. 2nd job: builds the application in a Docker image; the image is uploaded to DockerHub's *indigodatacloud* org

Proposal of INDIGO Continuous Deployment pipeline



1. Web application running on a INDIGO-reachable resource provider
2. A change in the codebase is committed in GitHub, which triggers job execution in Jenkins [user]
3. 1st job in the stack: tests/validates the source code, build the application, etc.
4. 2nd job: builds the application in a Docker image; the image is uploaded to DockerHub's *indigodatacloud* org
5. Submit the application to INDIGO platform

Proposal of INDIGO Continuous Deployment pipeline



1. Web application running on a INDIGO-reachable resource provider
2. A change in the codebase is committed in GitHub, which triggers job execution in Jenkins [user]
3. 1st job in the stack: tests/validates the source code, build the application, etc.
4. 2nd job: builds the application in a Docker image; the image is uploaded to DockerHub's *indigodatacloud* org
5. Submit the application to INDIGO platform
6. New change is automatically displayed

Adopting DevOps on user apps: benefits & improvements

INDIGO - DataCloud

Benefits

- No user interaction
  - Application developer only needs to deal with its code
  - Application will be automatically deployed using INDIGO platform
- New features/bugfixes are available straightaway
- Application reliability enhanced
  - Applying QA procedures

Lines of improvement

- Jenkins plugin to interface directly with the INDIGO PaaS orchestrator
  - Job definition would be simpler
- GitHub integrations: https://github.com/integrations
- DockerHub webhooks
  - React to automated builds of Docker images, triggering application submission to INDIGO platform, bypassing Jenkins (for apps not requiring testing)

Thanks for your attention

INDIGO - DataCloud

Questions?