

Build & Release Confidently with Continuous Integration & Delivery

Author: Nho Luong

Skill: DevOps Engineer Lead



The left screenshot shows the CertMetrics platform interface. It displays a grid of active certifications under the 'Certification status' section. The visible ones include:

- AWS Certified Solutions Architect - Professional (SAP)
- AWS Certified Security - Specialty (SCS)
- AWS Certified DevOps Engineer - Professional (DOP)
- AWS Certified Solutions Architect - Associate (SAA)

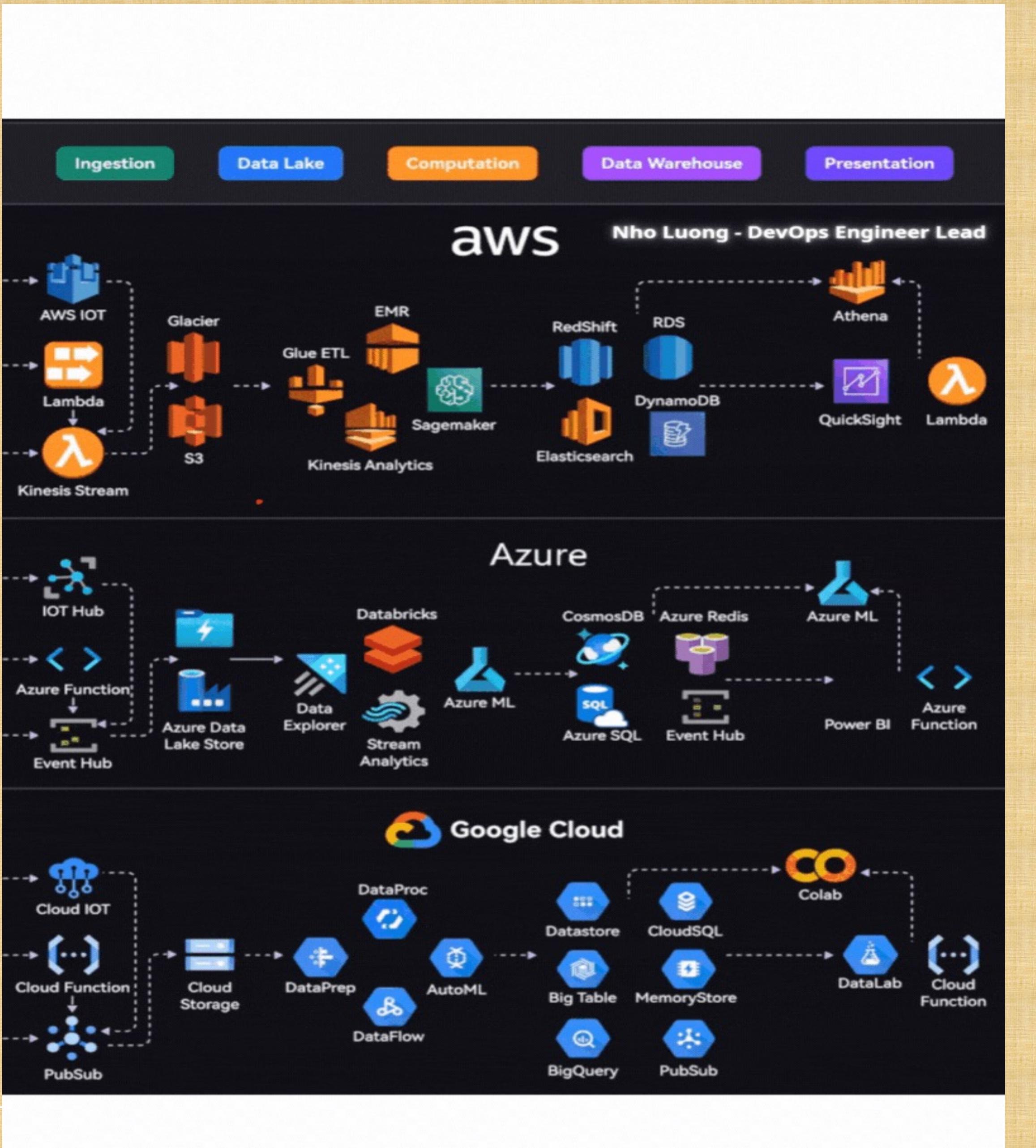
The right screenshot shows the Microsoft Learn user profile page. Under the 'Credentials' tab, it lists four active certifications:

- Microsoft Certified: DevOps Engineer Expert
- Microsoft Certified: Azure Solutions Architect Expert
- Microsoft Certified: Azure Administrator Associate
- Microsoft Certified: Azure Fundamentals

What is CI/CD?

Continuous Integration

- Run a series of scripts automatically, any time changes are
- pushed ~~Continuously Integrate~~ our changes Automated
- tests Coding standards Static code analysis ...et cetera
-
-
-



Dependencies & Artifacts

- Dependency management Anything
- that's generated for your app:
 - Compiled and/or minified
 - files Binaries
- **Remember:** source control should only worry about the source

Continuous Delivery

- Being able to deploy on-demand One-click deployments
- You can't forget a step if everything's automatic!



Continuous Delivery v Deployment

Delivery

Some manual step to deploy

Deployment

Always Be Deployin'

Continuous Delivery doesn't mean every change is deployed to production ASAP. It means every change is proven to be deployable at any time

3:25 AM - 28 Aug 2013

308 Retweets 165 Likes



12

308

165



Should I deploy on a
Friday at 5pm?

|

NO

|

what if I need to jus...

|

OMFG... NO

Deploy with Confidence!

**In order to deploy confidently, we must have confidence in
our tools.**

Setting up a CI/CD Pipeline

What is a Pipeline?

- A route from development to production
- Triggered by different events (push, merge, etc.)
- Different branches may take different paths



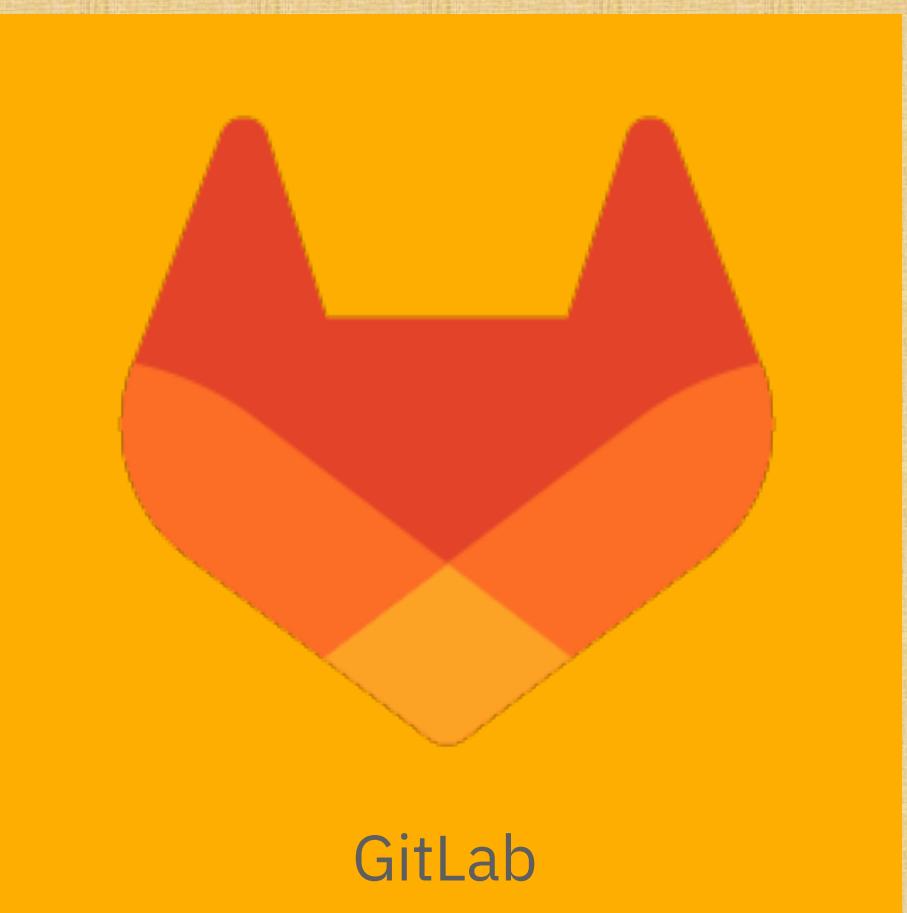
Photo by Christian Bass

Popular CI/CD

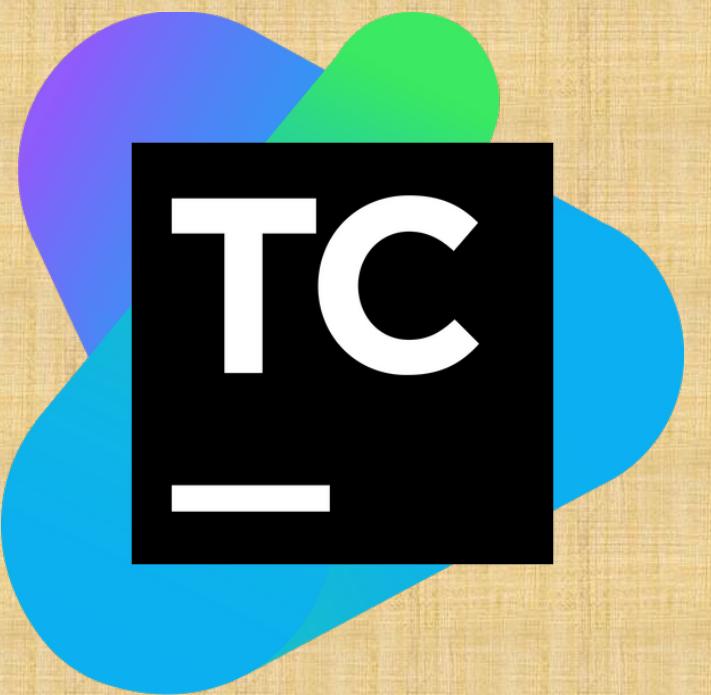
Providers



GitHub



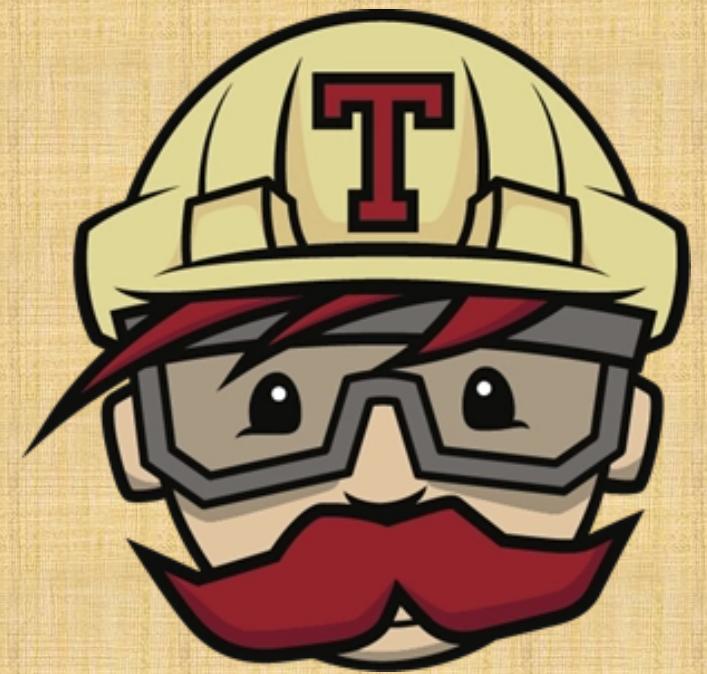
GitLab



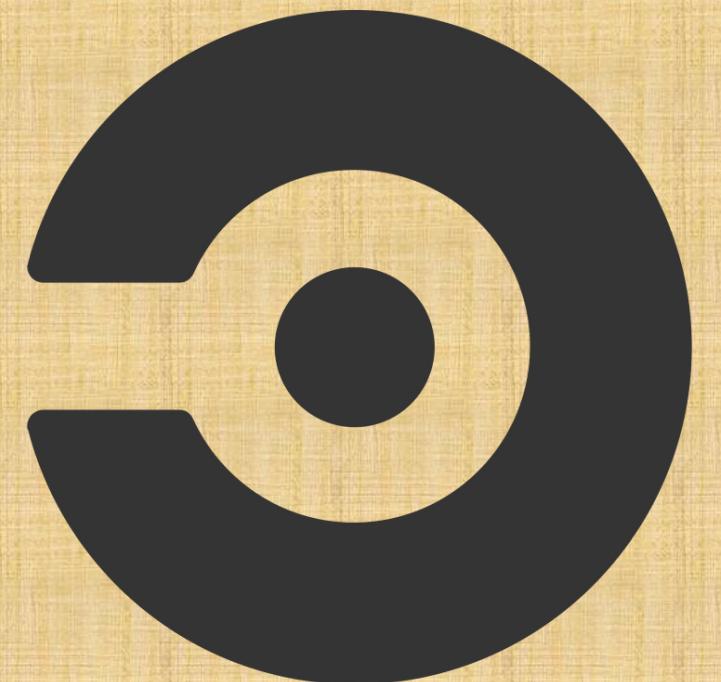
TeamCity



Jenkins



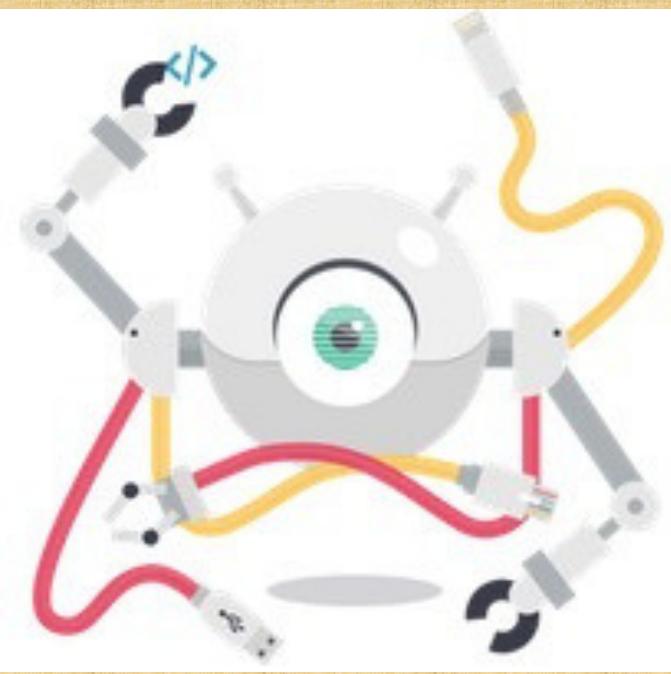
Travis
CI



CircleCI



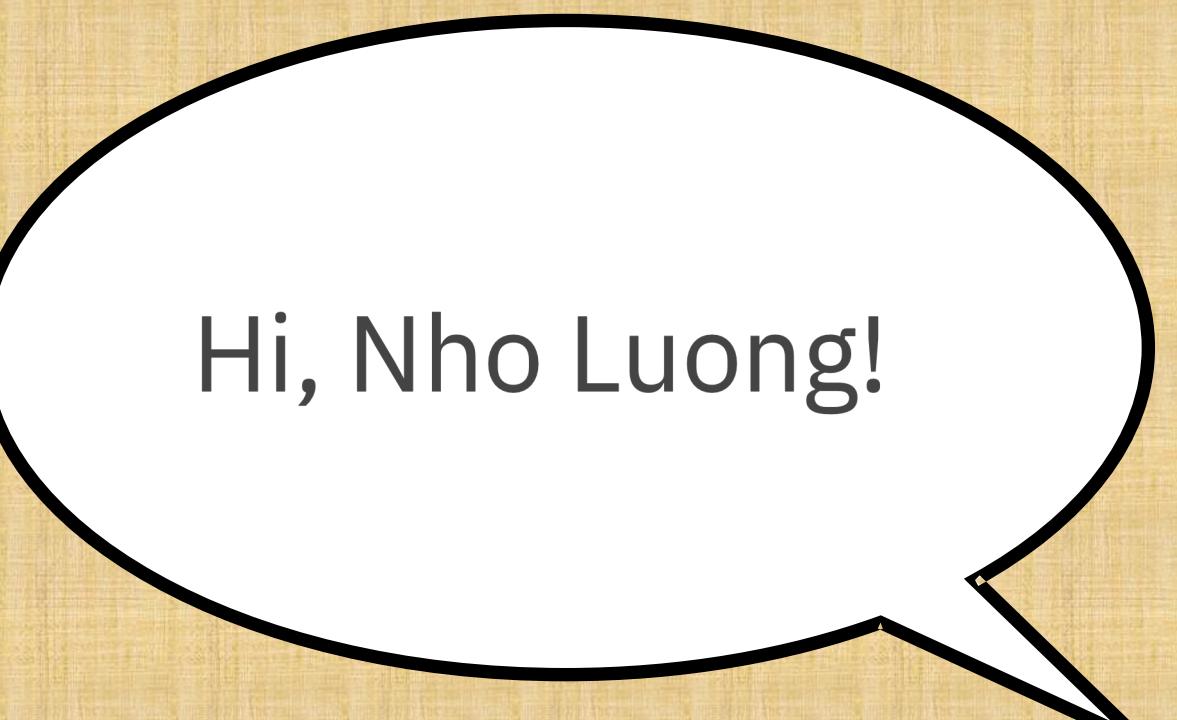
Codeship



Deploybot

Welcome to GitLab!

- Source code management tool Hosted platform or
- available for private installation Includes CI/CD
- tools!



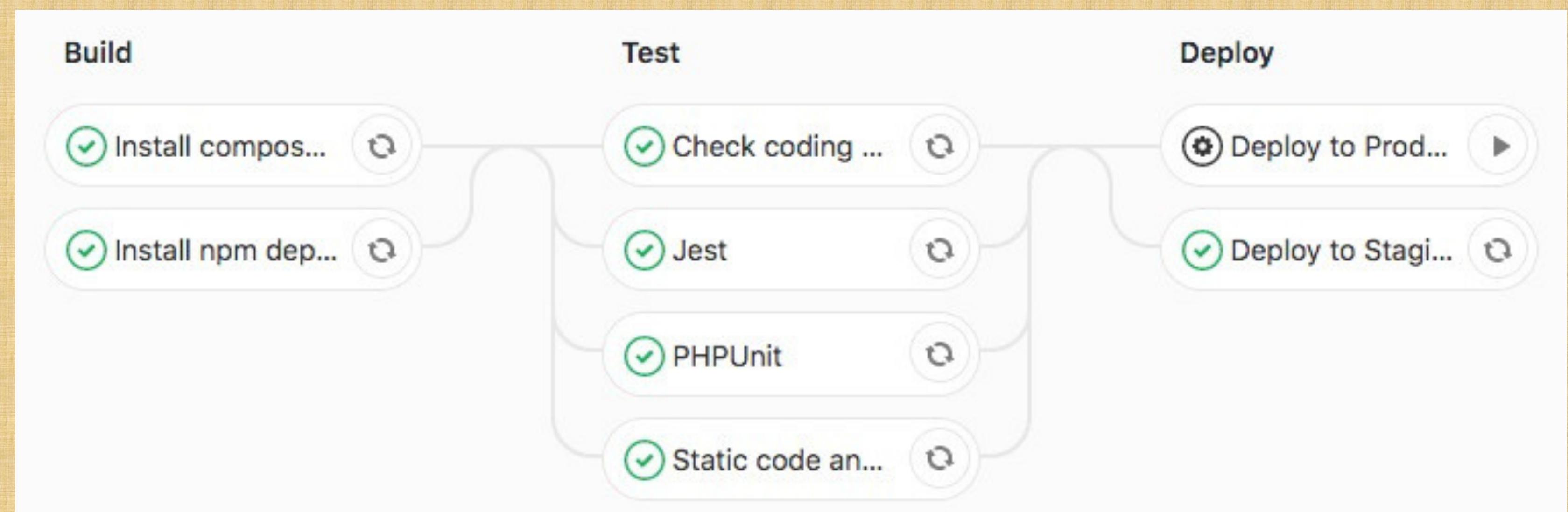
Hi, Nho Luong!

GitLab CI/CD Tools

- Define multiple stages (build, test, deploy, etc.) Each stage has one or more jobs
- Multiple runners == run jobs in parallel!

A Typical Pipeline

Configured in `.gitlab-ci.yml`



A Simple Job

Install npm dependencies:

stage: build

script:

- npm install --no-progress
- npm run prod

artifacts:

paths:

- public

cache:

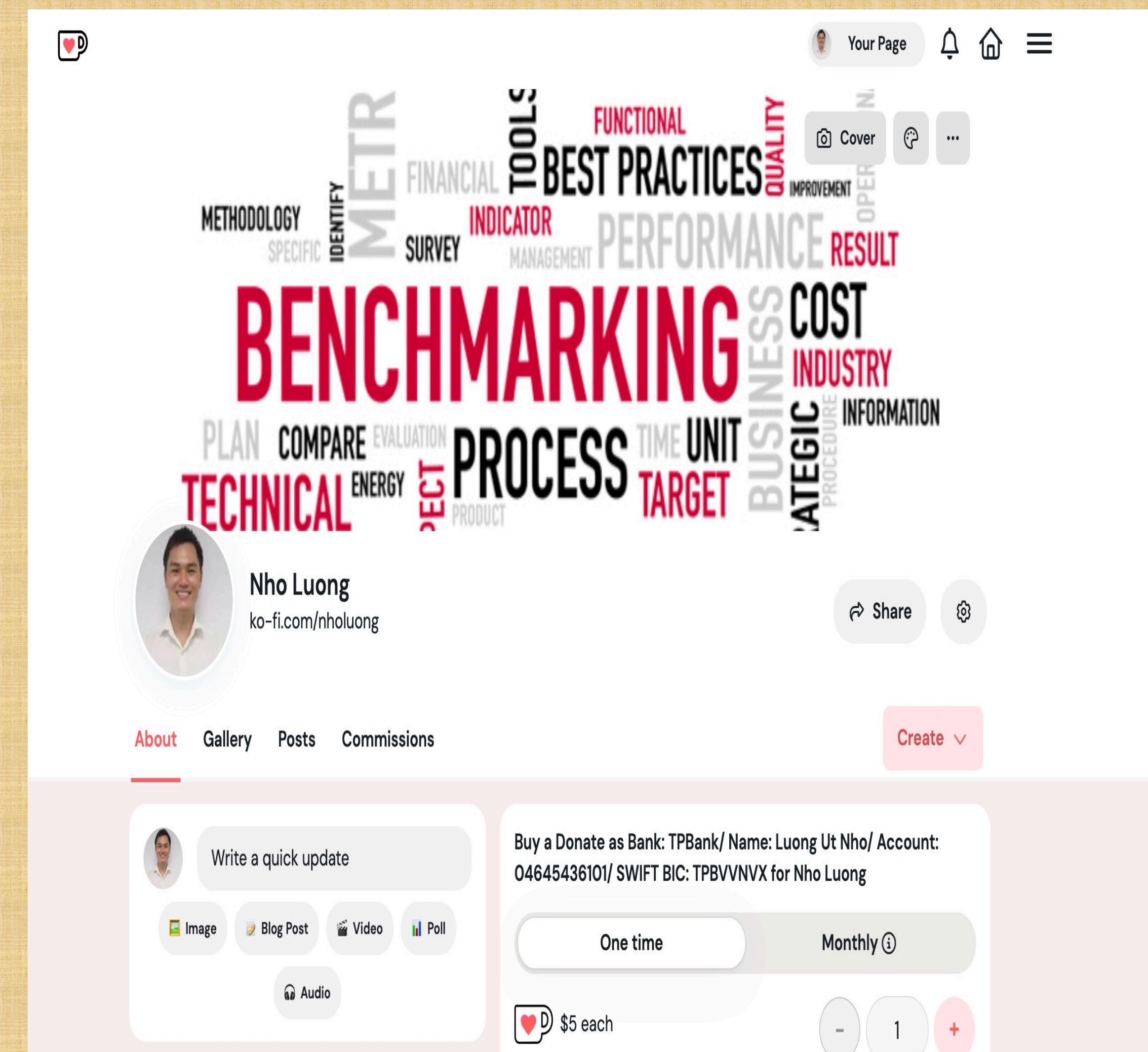
key:

files:

- package-lock.json

paths:

- node_modules



Highly Configurable

docs.gitlab.com/ee/ci/yaml

- Specify a base Docker image Determine conditions
- under which jobs should run Define dependencies
- between jobs Pre/post-script hooks Dynamic
- environments Automatically tag new releases
-
-

Environment Variables

Type	Key	Value	State	Masked	Scope
Variable	PRODUCTION_S	*****	Protected	<input type="button" value="x"/>	<input type="button" value="x"/>
Variable	PRODUCTION_T	*****	Protected	<input type="button" value="x"/>	<input type="button" value="x"/>
Variable	SSH_PRIVATE_K	*****	Protected	<input checked="" type="button" value="✓"/>	<input type="button" value="x"/>
Variable	STAGING_SERVE	*****	Protected	<input type="button" value="x"/>	<input type="button" value="x"/>
Variable	STAGING_TARGET	*****	Protected	<input type="button" value="x"/>	<input type="button" value="x"/>
Variable	TARGET_USER	*****	Protected	<input type="button" value="x"/>	<input type="button" value="x"/>

Simple Deployments

What Does our Deployment Process Look Like Right Now?

- (S)FTP? SSH +
- git
- pull? rsync? D
- ocker?

A Bare-Bones Deployment Method

1. Build the app
2. scp a tarball to production
3. rsync the files into the web root

Archive, scp, and rsync

ship_to_production:

stage: deploy

script:

```
- tar -czf release.tgz dist/* - scp release.tgz  
"${SSH_USER}@${PRODUCTION_SERVER}:/tmp" - ssh  
"${SSH_USER}@${PRODUCTION_SERVER}" "cd /tmp \  
  && tar -xzf release.tgz \  
  && rsync release/ /var/www/html/"
```

Atomic Deployments

releases/

Contains multiple, timestamped deployments

shared/

Data that should persist between deployments

current

Symlink to the current release in releases/, part of web root

Atomic Deployments

```
# /var/www releases/
```

```
+ 1563759801/ # Oldest +
1563759802/ +
1563759803/ # Newest      <= current symlink
shared/
+ logs/ +
uploads/ -
.env
```

Atomic Deployments

stevegrunwell.com/blog/atomic-deployments-from-scratch

ship_to_production:

 stage: deploy

 script:

```
# 1. Create + scp tarball # 2. Extract to  
/var/www/releases/{TIMESTAMP} # 3. Symlink  
shared assets # 4. Update the `current` symlink #  
5. Reload the web server # 6. (Optional) Roll off  
older releases
```

Extending Our Pipelines

Blue-Green Deployments

- Run multiple production environments/servers
 - Some active (blue), some idle (green)
 - Deploy to green
 - Once ready, route traffic to green
 - Blue becomes idle
 - In case of issues, re-route to blue



Automate Release Chores

Make your pipeline work for you!

- Automatically build Docker images for your app
- Generate + publish API documentation
- Code coverage reporting
- Webhooks If you can script it, you can do it!

Remember: This Isn't Magic!





Thank You