

Gitlab Auto-DevOps

CI-CD without breaking a sweat!

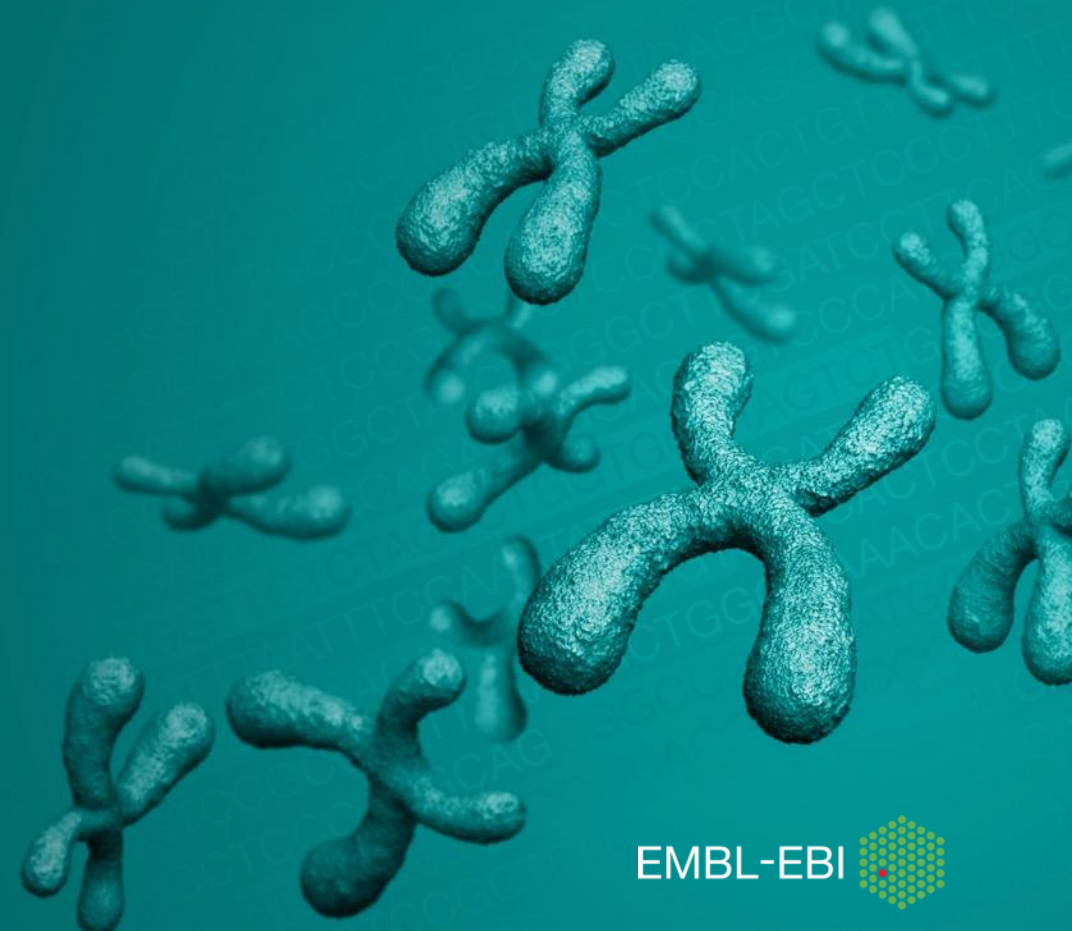
Soumyadip De

Cloud DevOps Engineer

Technology & Science Integration,

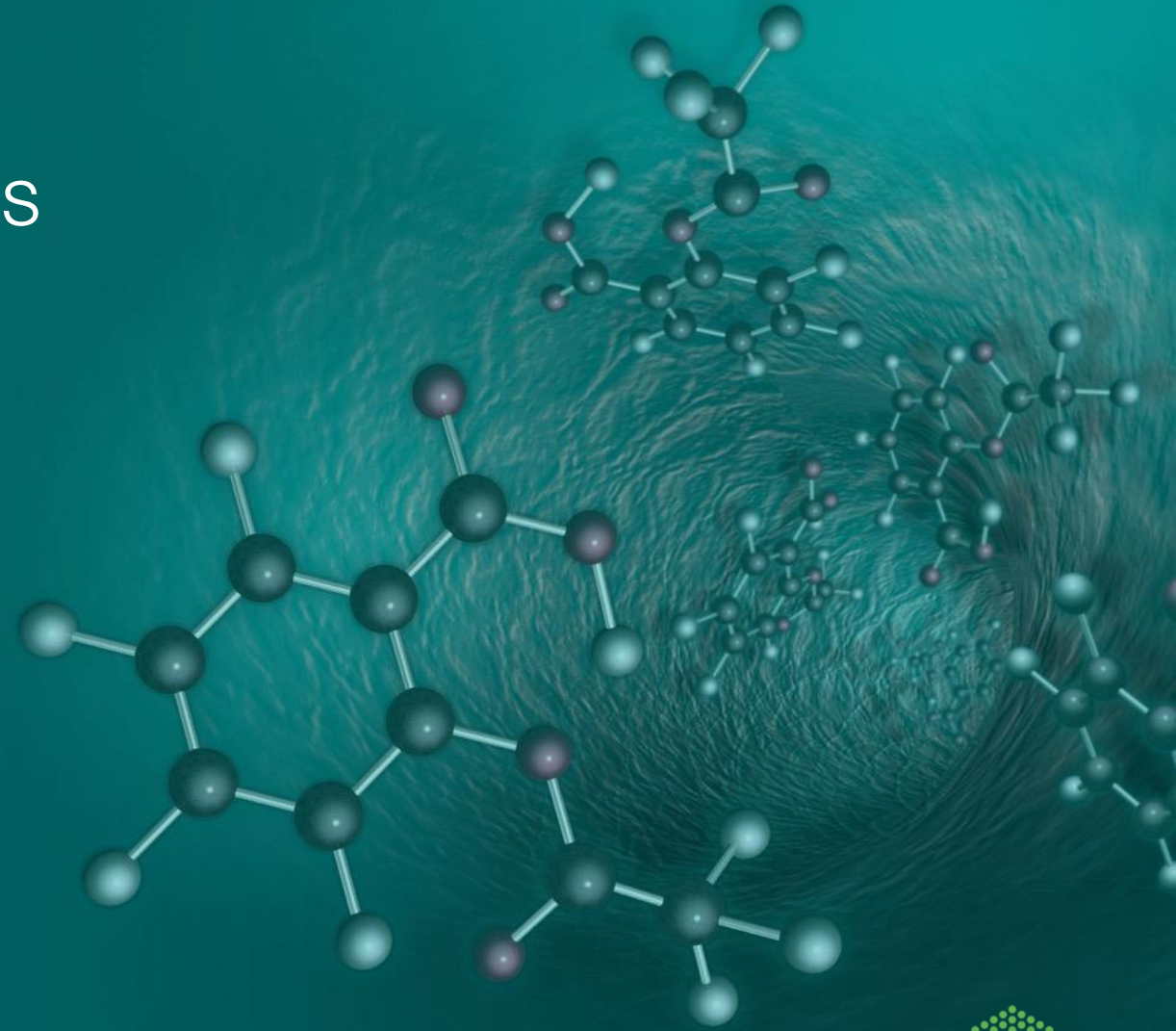
Technical Service Cluster

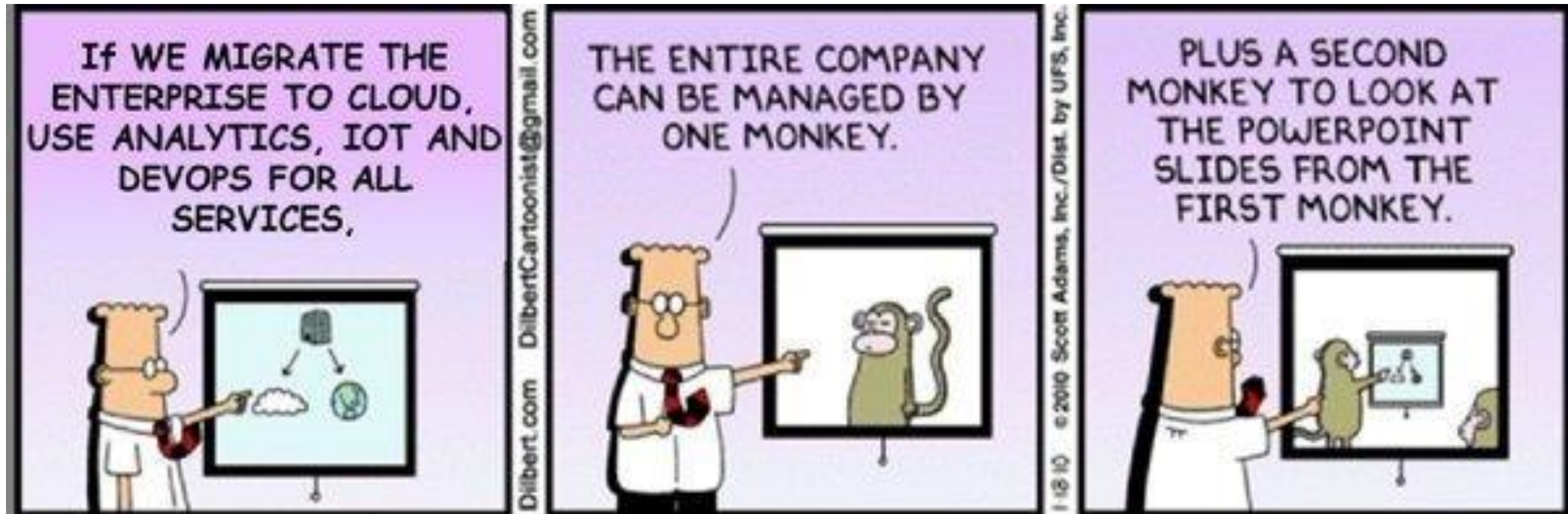
EMBL-EBI, Hinxton, UK



Gitlab Auto DevOps

- ❑ What is Gitlab Auto DevOps?
- ❑ Auto DevOps – pros and cons
- ❑ Pre-requisites
- ❑ How to use Auto DevOps
- ❑ Demo





IF WE MIGRATE THE ENTERPRISE TO CLOUD, USE ANALYTICS, IOT AND DEVOPS FOR ALL SERVICES,

THE ENTIRE COMPANY CAN BE MANAGED BY ONE MONKEY.

PLUS A SECOND MONKEY TO LOOK AT THE POWERPOINT SLIDES FROM THE FIRST MONKEY.

What is Gitlab Auto DevOps?

Gitlab Auto DevOps

- Pre-defined CI/CD configuration.
- Allows you to automatically detect, build, test, deploy, and monitor your applications.
- Leverages CI/CD best practices and tools.
- Simplifies the setup and execution of a mature & modern software development lifecycle.



Auto DevOps – pros and cons

Auto DevOps - Pros

- No scripting!
- Enables you to leverage DevOps best practices and tooling.
- Brings consistency on DevOps across projects

✓ passed

Auto DevOps - Cons

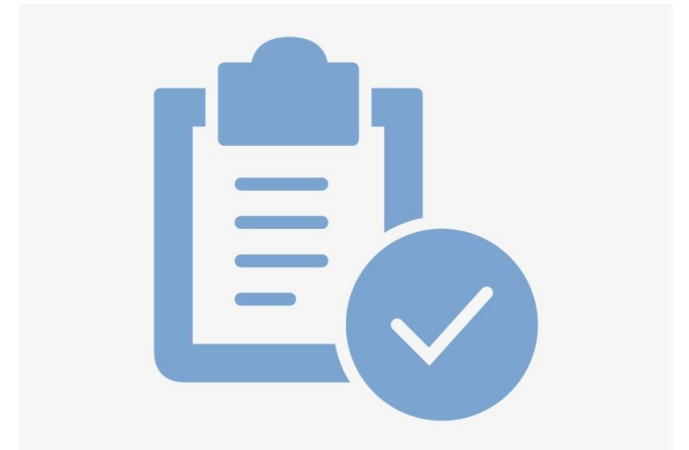
- Application needs to be Kubernetes-native
- May not be suitable for application dependant on multiple services
- May misbehave if the deployed application is managed outside gitlab

⊗ failed

Pre-requisites

Auto-DevOps pre-requisites

- GitLab Runner – if using gitlab.com it comes with shared runners
- Base domain – can use nip.io
- Kubernetes – can be provisioned from gitlab instance
- Application
 - Should be listening to 5000 tcp
 - Should be stateless and Kubernetes-native
 - Recommended to have Dockerfile



How to use Auto DevOps

How to use Auto DevOps

- Can be enabled in Project Setting > CI / CD > Auto-DevOps

Auto DevOps

Auto DevOps can automatically build, test, and deploy applications based on predefined continuous integration and delivery configuration. [Learn more about Auto DevOps](#) or use our [quick start guide](#) to get started right away.

☒ Default to Auto DevOps pipeline

The Auto DevOps pipeline will run if no alternative CI configuration file is found. [More information](#)

Add a [Kubernetes cluster integration](#) with a domain or create an `AUTO_DEVOPS_PLATFORM_TARGET` CI variable.

Deployment strategy

- ☒ Continuous deployment to production [?](#)
- ☐ Continuous deployment to production using timed incremental rollout [?](#)
- ☐ Automatic deployment to staging, manual deployment to production [?](#)

Save changes

How to use Auto DevOps

- Can be added as a template while editing .gitlab-ci.yml

Edit file

The screenshot shows the GitLab CI/CD editor interface. At the top, there are tabs for 'Write' and 'Preview changes'. Below the tabs, there is a dropdown menu for the branch, currently set to 'master'. Next to it is a text input field containing '.gitlab-ci.yml'. To the right of this is another dropdown menu labeled '.gitlab-ci.yml'. Further right is a dropdown menu labeled 'Apply a template'. To the right of this is a button labeled 'Soft wrap' and a dropdown menu labeled 'text'. The main area of the editor displays the content of the .gitlab-ci.yml file, which is a template for Auto DevOps. The content includes comments about building a Docker image, running tests, and creating a review app. The 'Apply a template' dropdown menu is open, showing a search bar with the text 'Au' and a list of templates. The templates are categorized into 'General', 'Pages', and 'Security'. Under 'General', there is a checked item 'Auto-DevOps' and an item 'Managed-Cluster-Applications'. Under 'Pages', there is an item 'SwaggerUI'. Under 'Security', there is an item 'Coverage-Fuzzing'.

Write Preview changes

master .gitlab-ci.yml .gitlab-ci.yml Apply a template Soft wrap text

```
1 # This file is a template, and might need editing before it works on your machine
2 # Auto DevOps
3 # This CI/CD configuration provides a standard pipeline for
4 # * building a Docker image (using a buildpack if necessary),
5 # * storing the image in the container registry,
6 # * running tests from a buildpack,
7 # * running code quality analysis,
8 # * creating a review app for each topic branch,
9 # * and continuous deployment to production
10 #
11 # Test jobs may be disabled by setting environment variables:
12 # * test: TEST_DISABLED
13 # * code_quality: CODE_QUALITY_DISABLED
14 # * license_management: LICENSE_MANAGEMENT_DISABLED
15 # * performance: PERFORMANCE_DISABLED
16 # * load_performance: LOAD_PERFORMANCE_DISABLED
17 # * sast: SAST_DISABLED
18 # * secret_detection: SECRET_DETECTION_DISABLED
19 # * dependency_scanning: DEPENDENCY_SCANNING_DISABLED
20 # * container_scanning: CONTAINER_SCANNING_DISABLED
21 # * dast: DAST_DISABLED
22 # * review: REVIEW_DISABLED
23 # * stop_review: REVIEW_DISABLED
24 #
```

Au

General

- ✓ Auto-DevOps
- Managed-Cluster-Applications

Pages

- SwaggerUI

Security

- Coverage-Fuzzing

Demo

Thanks, Our team.

