



www.cmind.io

Automatic generation of CI/CD pipelines

Alexandre-Xavier Labonté-Lamoureux

MASc Student

alexandre-xavier.labonte-lamoureux.1@ens.etsmtl.ca

Friday March 7th 2025

Kaloom-TELUS ÉTS IRC f2f Meeting



Alexandre-Xavier Labonté-Lamoureux, 2025

A little about me

- ~1 year at Arctiq (consulting)
 - Clients: Banks, Payment, Trading, Fintechs and video game companies
 - Certified GitLab Professional Services “Engineer”
 - Mostly been doing GitLab → GitHub migrations or improving pipelines
- Previously: 3 years at Ubisoft
 - SRE/ Platform team, mostly doing cybersecurity in a DevOps environment

The story of automatic CI/CD pipeline generation

www.emind.io

How we got there

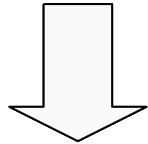
DevOps



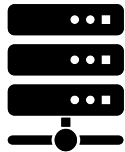
Developer



Software

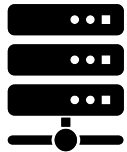
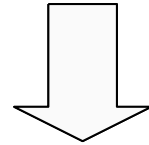
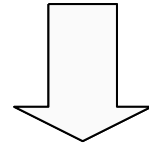
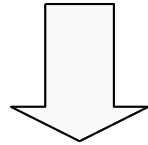
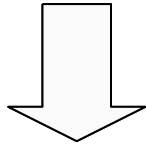
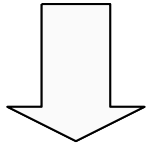


Provisioning and
CI/CD

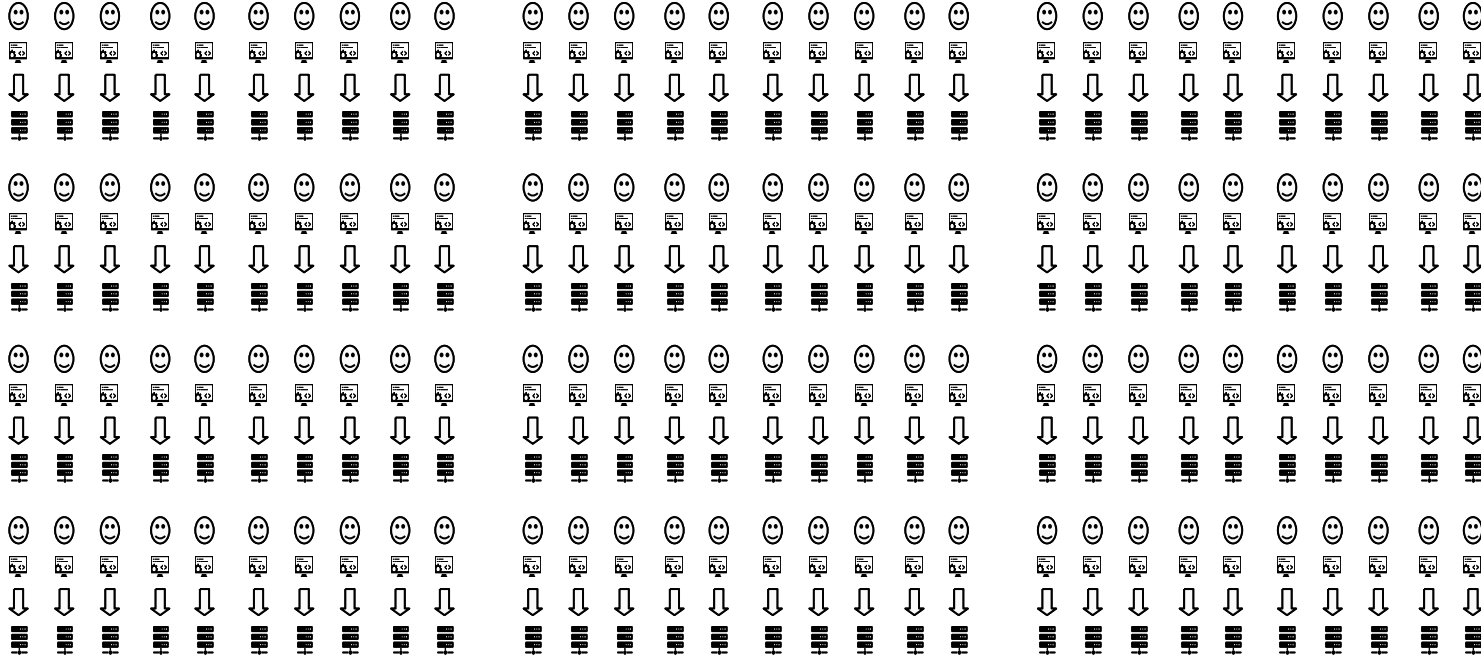


Infrastructure
(network, servers, etc.)

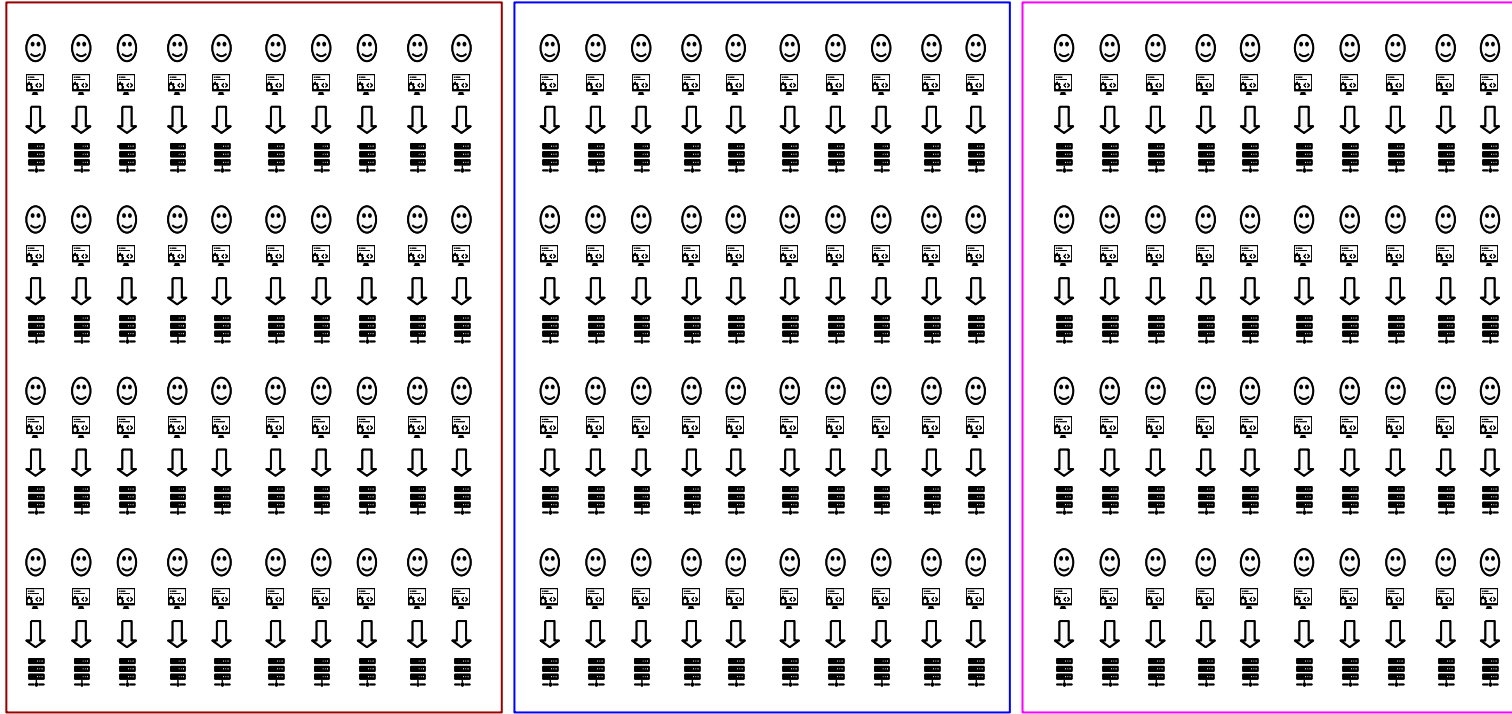
DevOps (at scale?)



DevOps at scale...

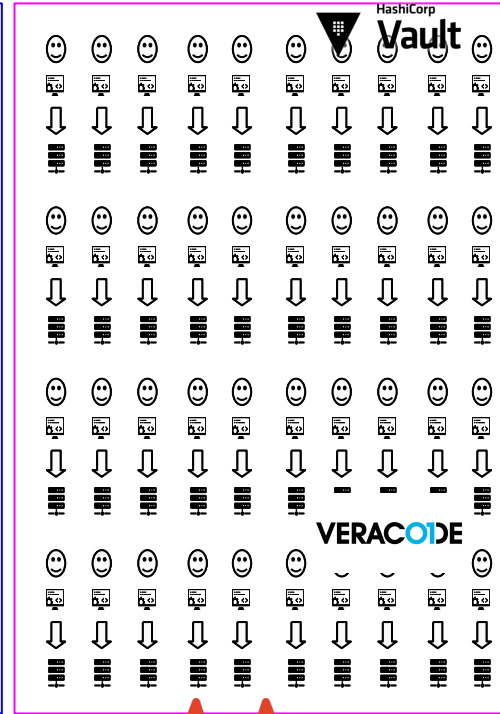
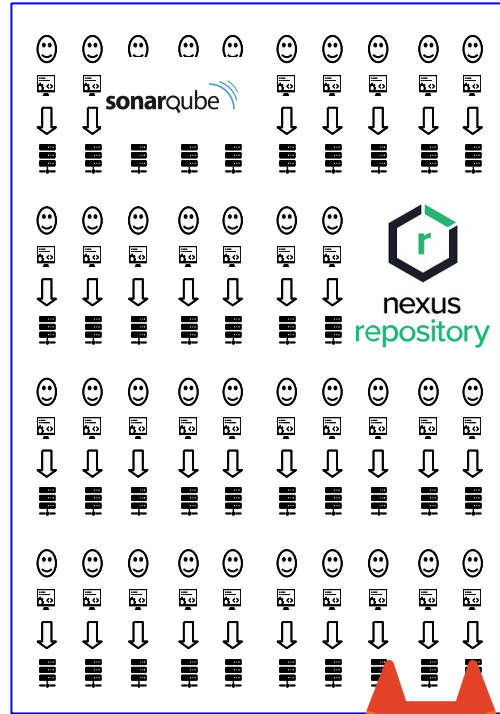
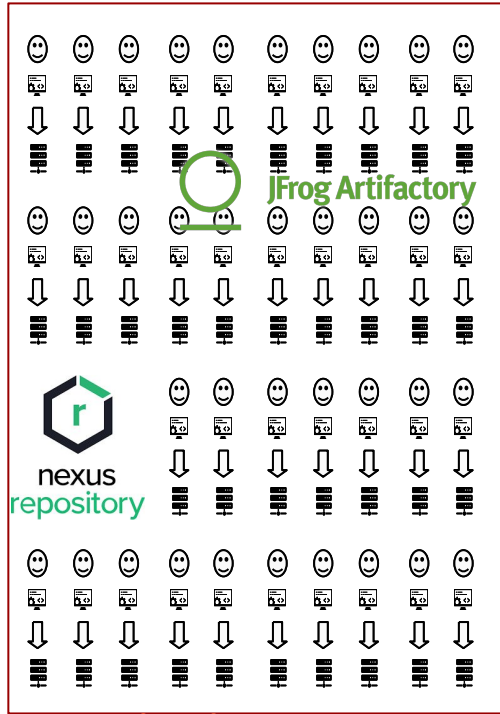


DevOps at scale...



with silos...

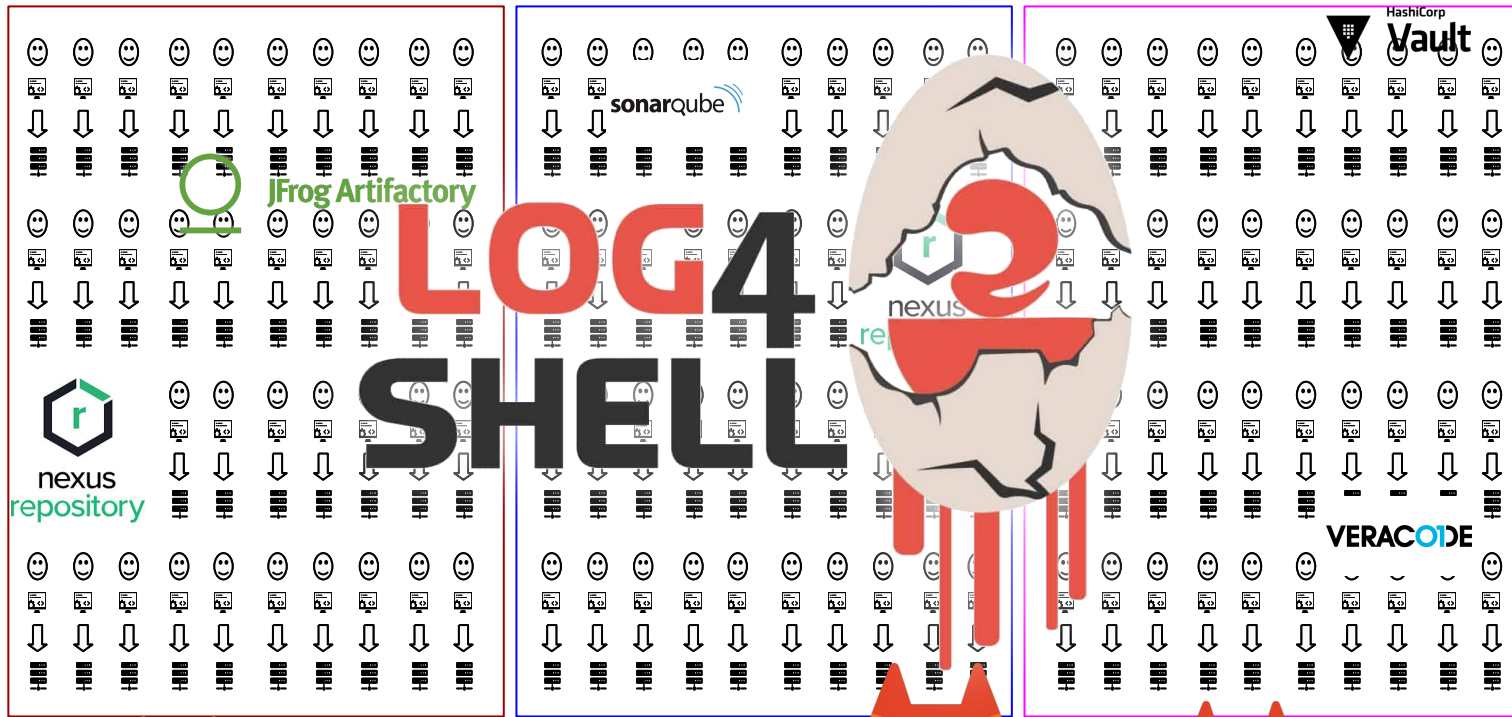
DevOps at scale...



with silos...

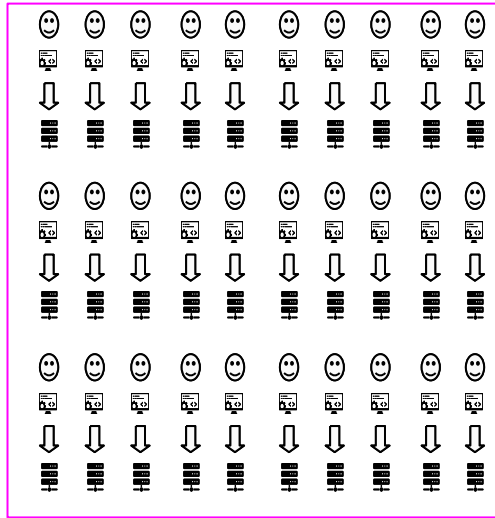
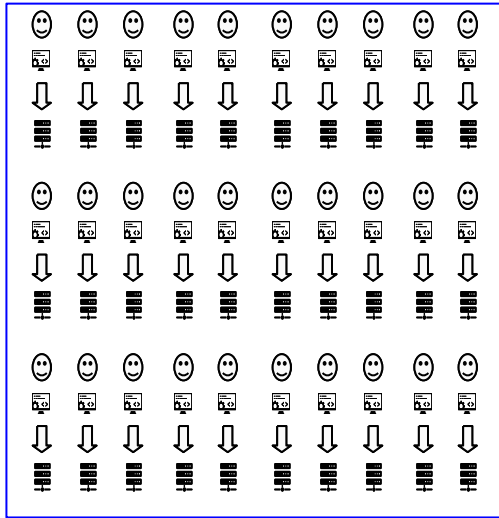
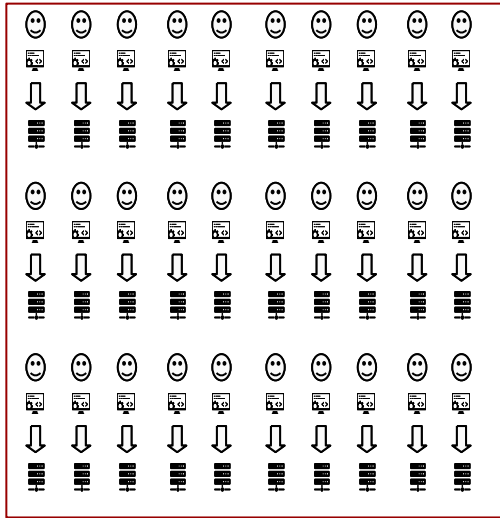
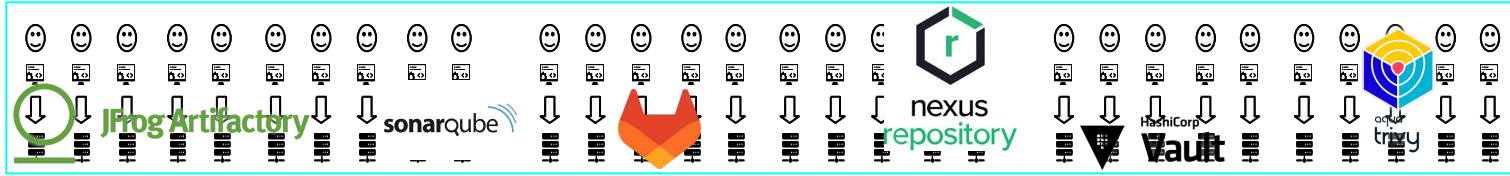


DevOps at scale...



with silos...

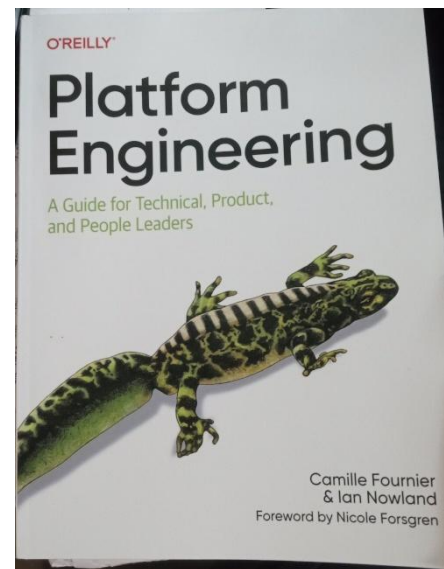
DevOps at scale...



with silos...

How to solve DevOps at scale with Silos

- Regroup common practices under a single team
 - Establish a **Platform Engineering** team
 - Define best practices (Golden paths / Paved roads)
 - Apply the Pareto principle
- Among everything, build a template engine for
 - CI/CD pipelines
 - Infrastructure provisioning code such as Terraform



Attacking CI/CD pipeline complexity

www.cmind.io

Templates as a solution (maybe?)

GitLab templates

 .gitlab-ci.yml  1.60 KIB

```
8  stages:
9  - build
10 - test
11 - review
12 - deploy
13
14 include:
15   - template: Jobs/Build.gitlab-ci.yml
16   - template: Jobs/SAST.gitlab-ci.yml
17   - template: Jobs/Secret-Detection.gitlab-ci.yml
18   - template: Jobs/Dependency-Scanning.gitlab-ci.yml
19   - template: Jobs/Container-Scanning.gitlab-ci.yml
20
21 gemnasium-dependency_scanning:
22   variables:
23     DS_REMEDIATE: "false"
24
```

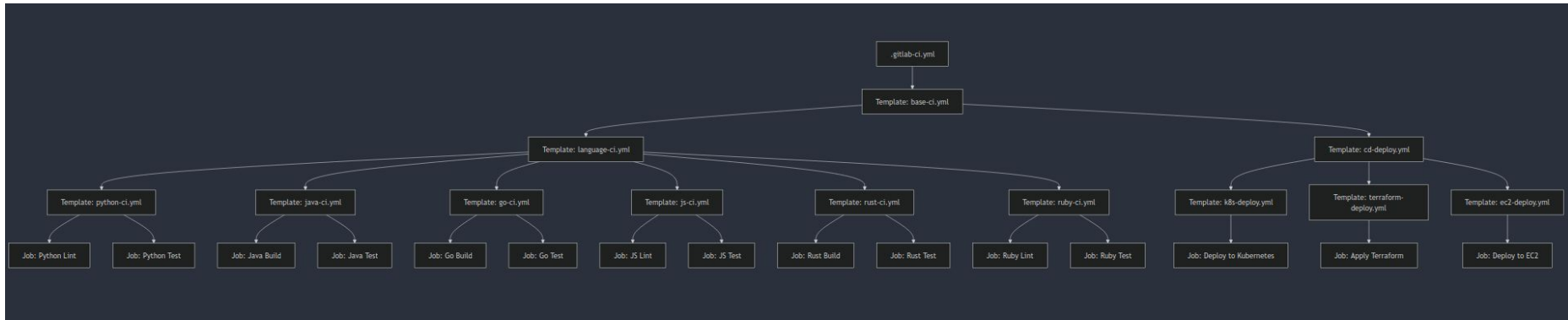
Ugh...

```
528 configure_system:
529   stage: configure
530   script:
531     - echo "Configuring the deployment system..."
532     - if [ "$ENVIRONMENT" == "production" ]; then
533       echo "Configuring for PRODUCTION environment...";
534       if [ "$BRANCH" == "main" ]; then
535         echo "Deploying from the MAIN branch...";
536         if [ "$DEPLOY_TYPE" == "full" ]; then
537           echo "Performing a FULL deployment...";
538           if [ "$CONFIRM_DEPLOY" == "true" ]; then
539             echo "Deployment confirmed. Proceeding with FULL production deployment...";
540             ./configure_production.sh --full;
541           else
542             echo "Deployment not confirmed. Aborting FULL production deployment.";
543           fi;
544         elif [ "$DEPLOY_TYPE" == "partial" ]; then
545           echo "Performing a PARTIAL deployment...";
546           if [ "$CONFIRM_DEPLOY" == "true" ]; then
547             echo "Deployment confirmed. Proceeding with PARTIAL production deployment...";
548             ./configure_production.sh --partial;
549           else
550             echo "Deployment not confirmed. Aborting PARTIAL production deployment.";
551           fi;
552         else
553           echo "Invalid DEPLOY_TYPE specified for production.";
554         fi;
555       else
556         echo "Deployments to PRODUCTION can only be done from the MAIN branch.";
557       fi;
558     elif [ "$ENVIRONMENT" == "staging" ]; then
559       echo "Configuring for STAGING environment...";
560       if [ "$BRANCH" == "develop" ]; then
561         echo "Deploying from the DEVELOP branch...";
562         if [ "$DEPLOY_TYPE" == "full" ]; then
563           echo "Performing a FULL staging deployment...";
564           ./configure_staging.sh --full;
565         elif [ "$DEPLOY_TYPE" == "partial" ]; then
566           echo "Performing a PARTIAL staging deployment...";
567           ./configure_staging.sh --partial;
568         else
569           echo "Invalid DEPLOY_TYPE specified for staging.";
570         fi;
571       else
572         echo "Deployments to STAGING can only be done from the DEVELOP branch.";
573       fi;
574     else
575       echo "Invalid ENVIRONMENT specified. Must be 'production' or 'staging'.";
576     fi
```

Ugh... (2)

```
643 deploy_system:
644   stage: deploy
645   script:
646     - echo "Deploying the system..."
647     - if [ "$ENVIRONMENT" == "production" ]; then
648         echo "Starting PRODUCTION deployment...";
649         if [ "$CONFIRM_DEPLOY" == "true" ]; then
650             echo "Deployment confirmed. Executing production deployment scripts...";
651             ./deploy_production.sh;
652         else
653             echo "Deployment not confirmed. Skipping production deployment.";
654         fi;
655     elif [ "$ENVIRONMENT" == "staging" ]; then
656         echo "Starting STAGING deployment...";
657         ./deploy_staging.sh;
658     else
659         echo "Invalid ENVIRONMENT specified. No deployment will be performed.";
660     fi
661     - echo "Deployment process completed."
```

Ugh... (illustrated)



The process of programming CI/CD pipelines

- Identify a problem or new feature to implement
- Make changes
- Wait for compute to become available
- Wait for compute to run the workload
- Review results of run
- Repeat

Outcome

- Low velocity
 - Slow iterations
 - Time lost waiting
 - Very likely to break something
 - Cannot test all possibilities (can we?)

Outcome

- Testing
 - Can't test them without running them
 - Need to test every possible scenarios
 - Each “if” block is two possibilities. 2^n where n is the number of possibilities.
 - Complexity grows very fast

Outcome

- Real life example @ Ubisoft
 - More than 1000 test cases
 - Each test case is an entire project that has a different setup
 - Requires a 24 runners farm and an entire day to go through all the tests
 - Tests are executed inside GitLab CI on real runners

Outcome

- Real life example @ Ubisoft
 - Approx 2 weeks required for a change
 - Approx 2 months required for contributions from external teams

Pipeline generator internals

www.emind.io

Template engine as the ultimate solution

Project structure

```

  ▾ PIPELINE-GENERATOR
    ▾ ci-templates
      ! docker.yml
      ! go.yml
      ! java.yml
      ! python.yml
    ▾ cmd / pipeline-generator
      -go config.go
      -go docker.go
      -go golang.go
      -go java.go
      -go main.go
      🐾 .gitlab-ci.yml
      ≡ go.mod
      ≡ go.sum
      ⓘ README.md

```

Project structure: templates

```
ci-templates > ! go.yml
1  image: golang:latest
2
3  stages:
4    - build
5    - test
6    - quality
7    - security
8
9  variables:
10  GOPATH: "$HOME/go"
11  BINARY_NAME: "app" # default binary name, should be detected automatically
12
13  cache:
14    paths:
15      - "$GOPATH/pkg/mod"
16
17  build:
18    stage: build
19    script:
20      - go mod download
21      - go build -o $BINARY_NAME ./cmd/*
22    artifacts:
23      paths:
24        - $BINARY_NAME
25      expire_in: 1 week
26      name: "${CI_PROJECT_NAME}-${CI_COMMIT_REF_NAME}"
```

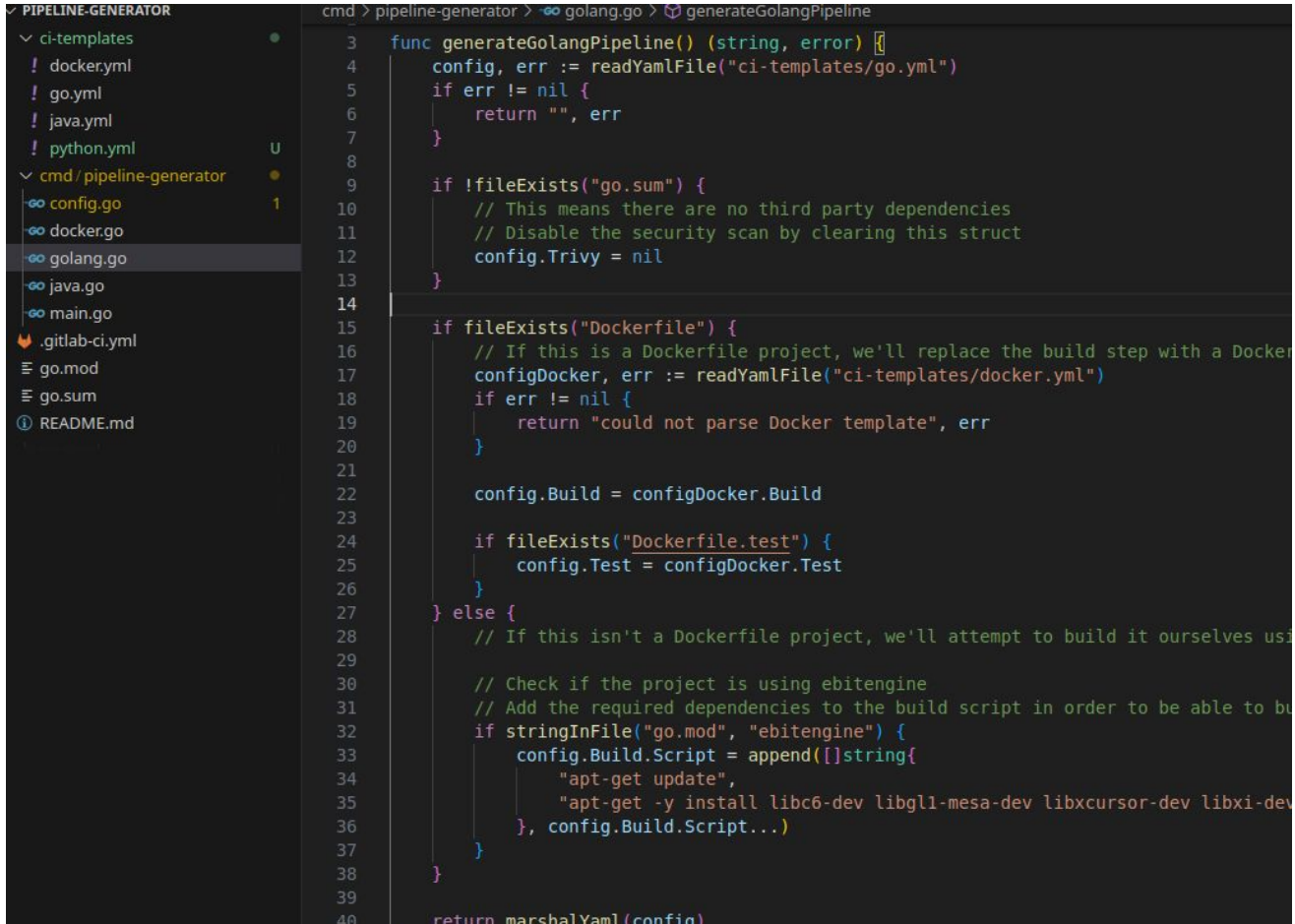
Project structure: yaml parser

```
cmd > pipeline-generator > -o config.go > ...
1  package main
2
3  import (
4      "fmt"
5      "os"
6
7      "gopkg.in/yaml.v3"
8  )
9
10 type Config struct {
11     Image      string          `yaml:"image"`
12     Stages     []string       `yaml:"stages"`
13     Variables  map[string]string `yaml:"variables"`
14     Cache      struct {
15         Paths []string `yaml:"paths"`
16     } `yaml:"cache"`
17     Services []string `yaml:"services,omitempty"`
18     Build    struct {
19         Image      string          `yaml:"image,omitempty"`
20         Stage      string         `yaml:"stage"`
21         Variables  map[string]string `yaml:"variables,omitempty"`
22         Services   []string       `yaml:"services,omitempty"`
23         Script     []string       `yaml:"script"`
24         Artifacts  struct {
25             Paths []string `yaml:"paths,omitempty"`
26             ExpireIn string `yaml:"expire_in,omitempty"`
27             Name string `yaml:"name,omitempty"`
28         } `yaml:"artifacts"`
29         Only []string `yaml:"only,omitempty"`
30     } `yaml:"build"`
31     Test *struct {
32         Image string          `yaml:"image,omitempty"`
33         Stage string         `yaml:"stage"`
```

Project structure: language detection

```
29
30 func main() {
31     if len(os.Args) < 2 {
32         log.Fatalln("Missing argument")
33     }
34
35     outputFile := os.Args[1]
36
37     if fileExists("go.mod") {
38         log.Println("Generating pipeline for Go project")
39         script, err := generateGolangPipeline()
40         if err != nil {
41             log.Fatalf("Error generating pipeline: %v", err)
42         }
43         os.WriteFile(outputFile, []byte(script), 0644)
44     } else if fileExists("pom.xml") {
45         log.Println("Generating pipeline for Java Maven project")
46         script, err := generateJavaMavenPipeline()
47         if err != nil {
48             log.Fatalf("Error generating pipeline: %v", err)
49         }
50         os.WriteFile(outputFile, []byte(script), 0644)
51     } else if fileExists("requirements.txt") {
52         log.Println("Generating pipeline for Python project")
53         script, err := generatePythonPipeline()
54         if err != nil {
55             log.Fatalf("Error generating pipeline: %v", err)
56         }
57         os.WriteFile(outputFile, []byte(script), 0644)
58     } else if fileExists("Dockerfile") {
59         log.Println("Generating pipeline for Docker project")
60         script, err := generateDockerPipeline()
61         if err != nil {
62             log.Fatalf("Error generating pipeline: %v", err)
63         }
64         os.WriteFile(outputFile, []byte(script), 0644)
65     } else {
```

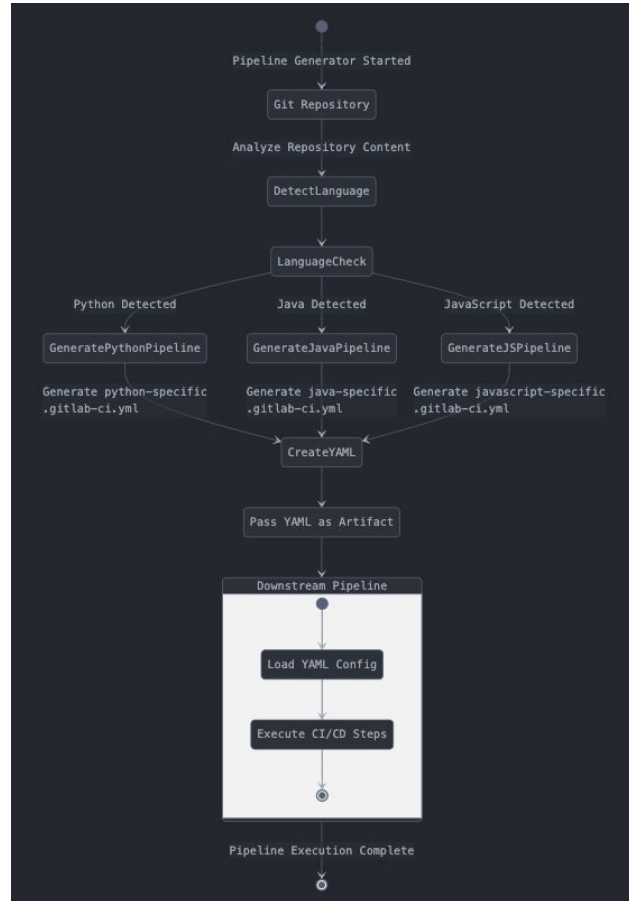
Project structure: language specific code



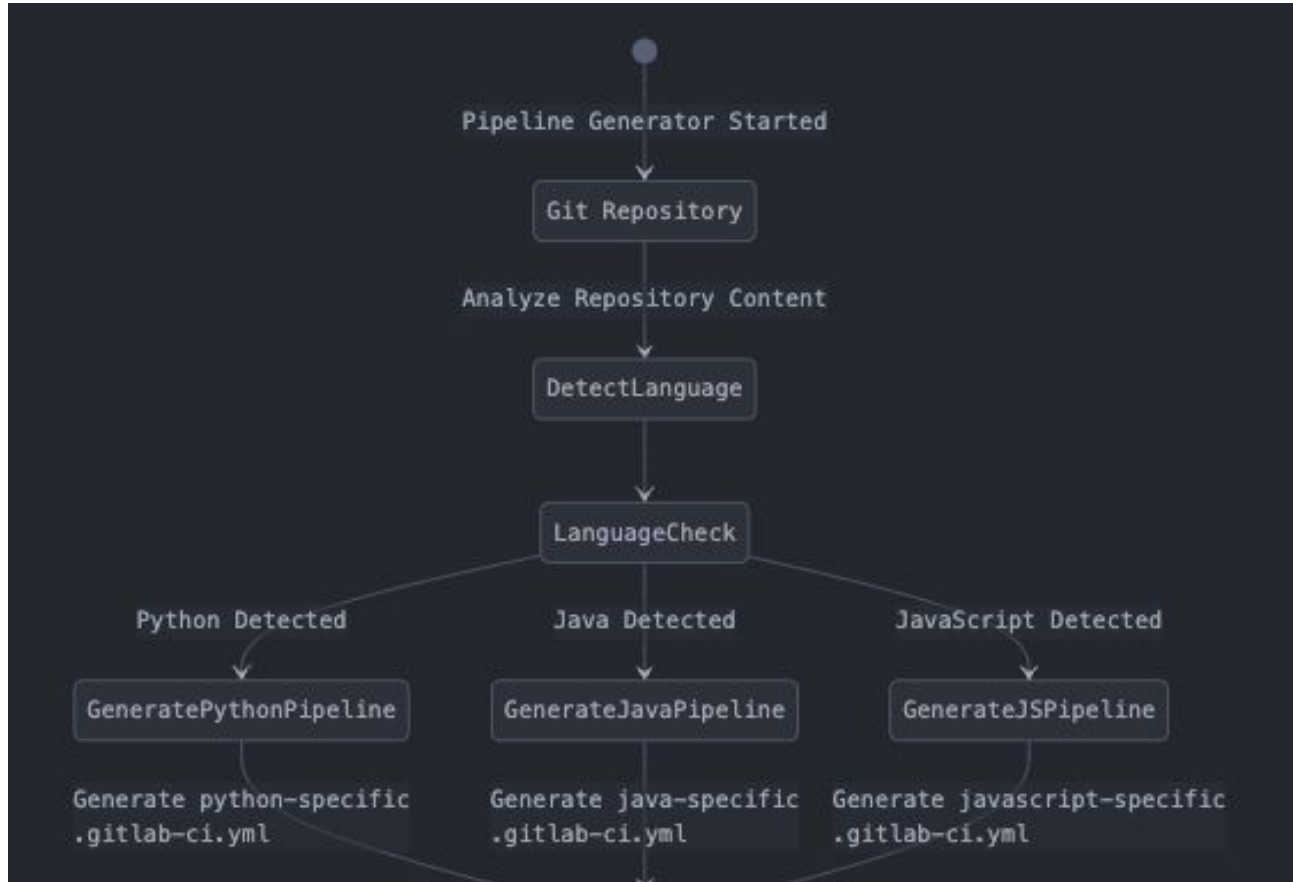
The image shows a code editor interface. On the left, a file explorer displays the project structure under the name 'PIPELINE-GENERATOR'. The files listed are: 'ci-templates' (a folder), 'docker.yml', 'go.yml', 'java.yml', 'python.yml', 'cmd/pipeline-generator' (a folder), 'config.go', 'docker.go', 'golang.go' (selected), 'java.go', 'main.go', '.gitlab-ci.yml', 'go.mod', 'go.sum', and 'README.md'. On the right, the code editor shows the content of 'golang.go'. The code is a Go function named 'generateGolangPipeline' that takes a string and an error pointer as arguments. It reads a YAML file 'ci-templates/go.yml' and checks for the existence of 'go.sum' and 'Dockerfile'. Depending on the presence of these files, it configures the build process, including setting 'Trivy' to nil, reading 'docker.yml' for Dockerfile projects, and checking for 'ebitengine' dependencies in 'go.mod' for non-Dockerfile projects. The function returns a marshaled YAML configuration.

```
cmd > pipeline-generator > golang.go > generateGolangPipeline
3 func generateGolangPipeline() (string, error) {
4     config, err := readYamlFile("ci-templates/go.yml")
5     if err != nil {
6         return "", err
7     }
8
9     if !fileExists("go.sum") {
10        // This means there are no third party dependencies
11        // Disable the security scan by clearing this struct
12        config.Trivy = nil
13    }
14
15    if fileExists("Dockerfile") {
16        // If this is a Dockerfile project, we'll replace the build step with a Docker
17        configDocker, err := readYamlFile("ci-templates/docker.yml")
18        if err != nil {
19            return "could not parse Docker template", err
20        }
21
22        config.Build = configDocker.Build
23
24        if fileExists("Dockerfile.test") {
25            config.Test = configDocker.Test
26        }
27    } else {
28        // If this isn't a Dockerfile project, we'll attempt to build it ourselves using
29
30        // Check if the project is using ebitengine
31        // Add the required dependencies to the build script in order to be able to bu
32        if stringInFile("go.mod", "ebitengine") {
33            config.Build.Script = append([]string{
34                "apt-get update",
35                "apt-get -y install libc6-dev libglib2.0-dev libxcursor-dev libxi-dev
36            }, config.Build.Script...)
37        }
38    }
39
40    return marshalYaml(config)
```

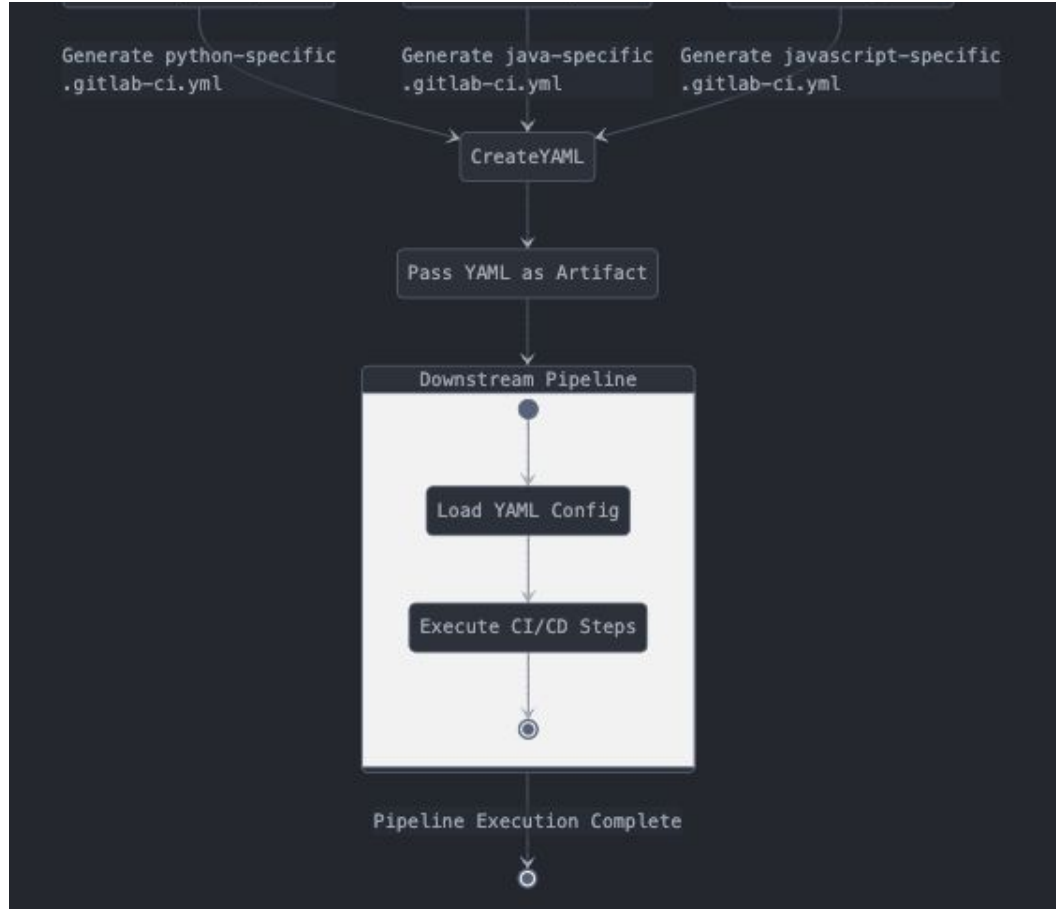
Pipeline generator workflow



Pipeline generator workflow




Pipeline generator workflow




Demo



www.emind.io


Demo - Empty test project

T **Test project**  Free

 main test-project / +

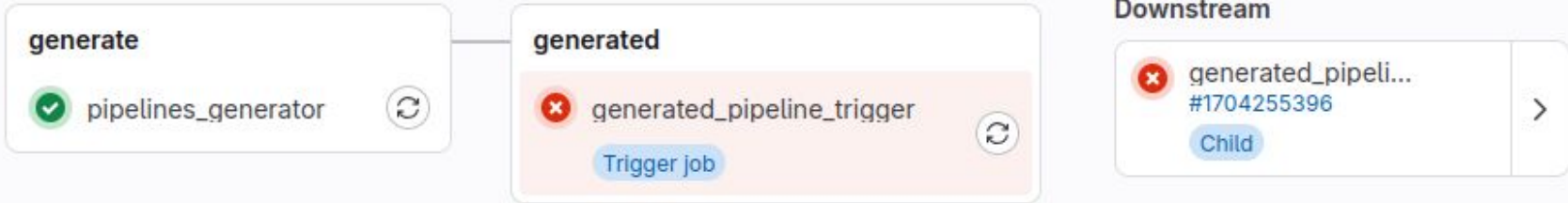
Find file Edit Code

 **remove files**
Alexandre-Xavier authored 52 seconds ago ✖ 007020ad  History

Name	Last commit	Last update
 .gitlab-ci.yml	test	3 minutes ago

Demo - CI script in the empty project

Pipeline Jobs 2 Tests 0



Demo - CI script in the empty project

```
 .gitlab-ci.yml  366 B
```

```
1 stages:
2   - generate
3   - generated
4
5 pipelines_generator:
6   stage: generate
7   image: axdoomer/pipeline-generator:latest
8   script:
9     - echo "Ran"
10  artifacts:
11    paths:
12      - generated-pipeline.yml
13
14  generated_pipeline_trigger:
15    stage: generated
16    trigger:
17      include:
18        - artifact: generated-pipeline.yml
19          job: pipelines_generator
20    strategy: depend
21
```

Demo - Go project has been committed

test-project

Find file

Edit ▾

Code ▾



test

Alexandre-Xavier authored 15 minutes ago

c8feab11



History

Name	Last commit	Last update
assets	test	15 minutes ago
cmd/maze-game	test	15 minutes ago
internal	test	15 minutes ago
.gitignore	test	15 minutes ago
.gitlab-ci.yml	test	15 minutes ago
README.md	test	15 minutes ago
go.mod	test	15 minutes ago
go.sum	test	15 minutes ago

Demo - The CI pipeline has been changed

Fix artifacts

✓ Passed Alexandre-Xavier created pipeline for commit `ebf0cd0e` 30 minutes ago, finished 27 minutes ago

For `main`

latest branch @ 2 jobs 0.71 2 minutes 18 seconds, queued for 0 seconds

Pipeline Jobs 2 Tests 0



Demo - Replacing the project with another

test-project

Find file

Edit ▾

Code ▾



test

Alexandre-Xavier authored 18 minutes ago


c8feab11





History





Name	Last commit	Last update
assets	test	18 minutes ago
cmd/maze-game	test	18 minutes ago
internal	test	18 minutes ago
.gitignore	test	18 minutes ago
.gitlab-ci.yml	test	18 minutes ago
README.md	test	18 minutes ago
go.mod	test	18 minutes ago
go.sum	test	18 minutes ago

Demo - Replaced project with Java project

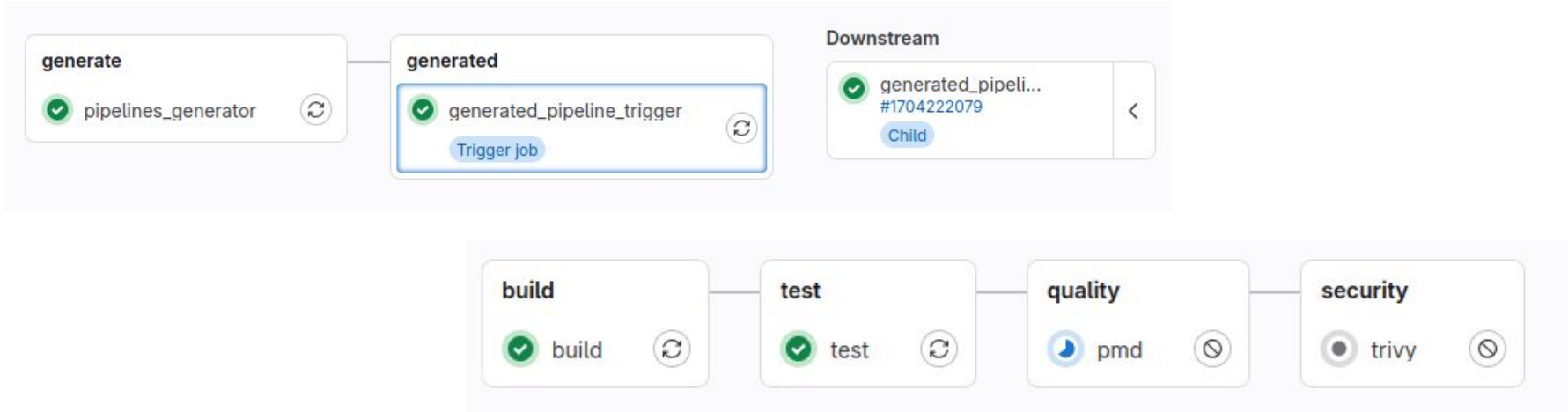
T **Test project**  Free

main ▾ test-project / + ▾ Find file Edit ▾ Code ▾



 **Replace Go project with Java project**
Alexandre-Xavier authored just now 8d3b14ad  History


Name	Last commit	Last update
src	Replace Go project with Java project	just now
 .gitignore	Replace Go project with Java project	just now
 .gitlab-ci.yml	Replace Go project with Java project	just now
 README.md	Replace Go project with Java project	just now
 pom.xml	Replace Go project with Java project	just now



Demo - Replaced project with Java project








Demo - Replaced project with Java project

T **Test project**  Free 

 main test-project / + Find file Edit Code

 **Replace Go project with Java project**  8d3b14ad History
Alexandre-Xavier authored 3 minutes ago

Name	Last commit	Last update
 src	Replace Go project with Java project	3 minutes ago
 .gitignore	Replace Go project with Java project	3 minutes ago
 .gitlab-ci.yml	Replace Go project with Java project	3 minutes ago
 README.md	Replace Go project with Java project	3 minutes ago
 pom.xml	Replace Go project with Java project	3 minutes ago

Demo - Replacing the project with a Docker project

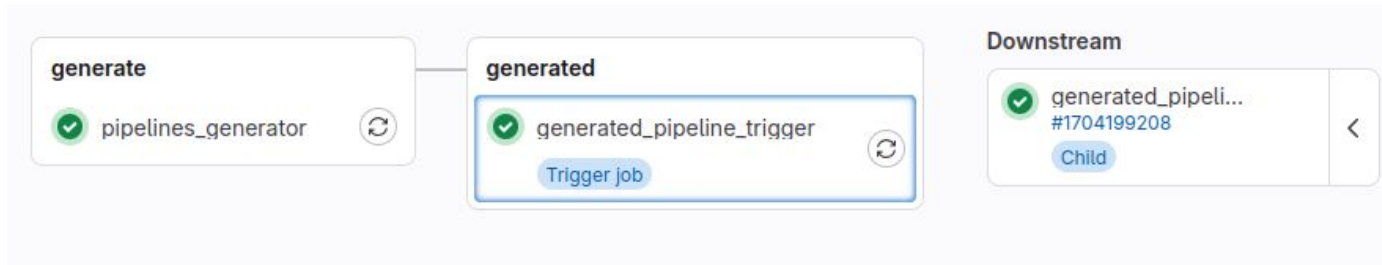
T **Test project** Free

main test-project / + Find file Edit Code

replace java project with bash docker project 27632a96 History
Alexandre-Xavier authored just now

Name	Last commit	Last update
.gitlab-ci.yml	replace java project with bash docker p...	just now
Dockerfile	replace java project with bash docker p...	just now
Dockerfile.test	replace java project with bash docker p...	just now
README.md	replace java project with bash docker p...	just now
build.sh	replace java project with bash docker p...	just now
test.sh	replace java project with bash docker p...	just now

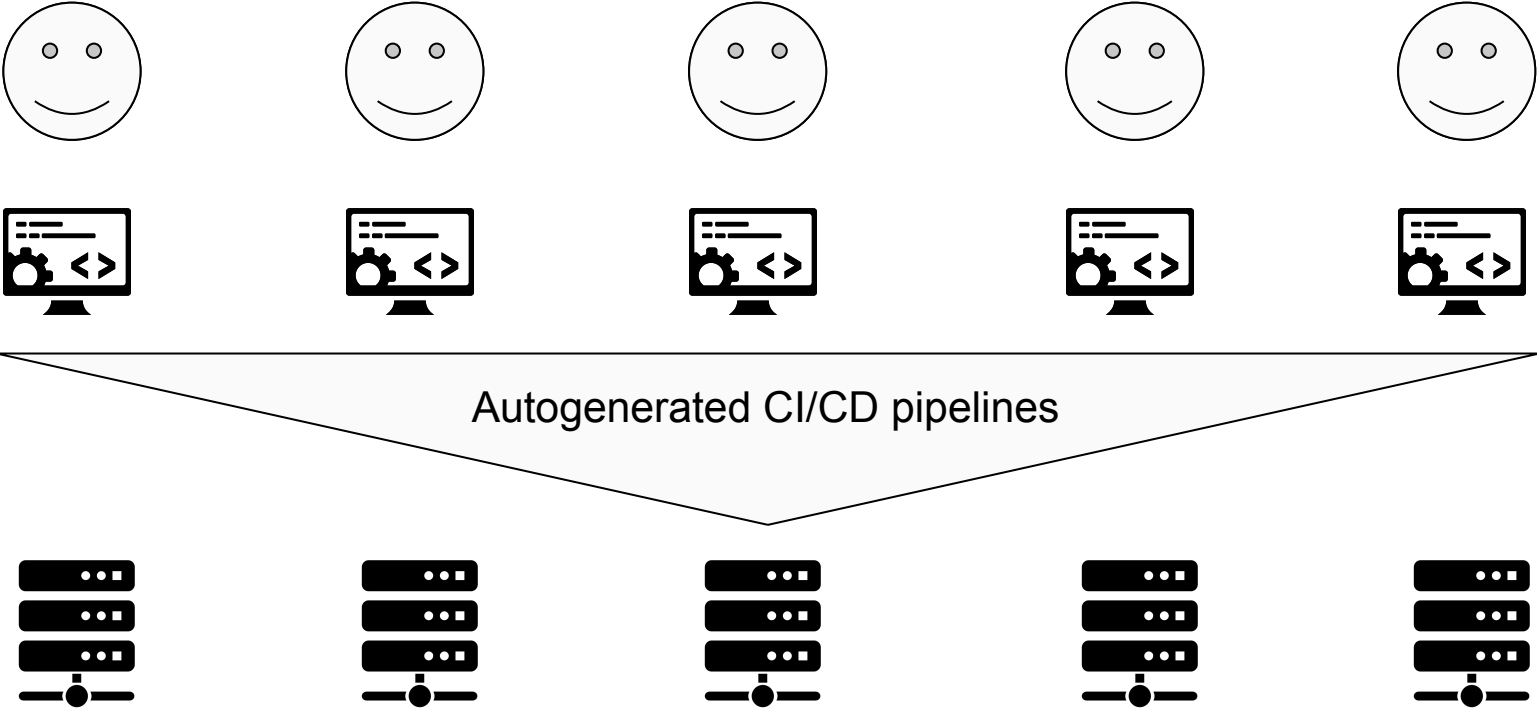
Demo - Docker CI pipeline



Advantages of this approach

- Contrary to a GitLab CI script, the generator is pure software
 - It can be unit tested
 - It can be integration tested
 - TDD (Test-driven development) can be used to develop pipelines
 - Less cognitive load for developers (better DevEx)
 - Pipelines are always up-to-date with the best practices

DevOps supported by PE team





Paper on Automatic Pipeline Provisioning



https://axdoomer.gitlab.io/Automatic_Pipeline_Provisioning.pdf

Send questions and feedback to my email