

DevOps

More than just YAML

Holland Gibson

1

SLOW DOWN

ADMIN

Billable to:

- FED portfolio (under Michelle H.)
- COMM/INTL portfolio (under Joel M.)
- PROFDEV

Who am I?

3

1. optix-edge fed team
2. worked as a DevOps engineer previously
3. @ me

Who am I?

Optix-Edge Fed

4

1. optix-edge fed team
2. worked as a DevOps engineer previously
3. @ me

Who am I?

Optix-Edge Fed

@ghgibson

5

1. optix-edge fed team
2. worked as a DevOps engineer previously
3. @ me

DevOps

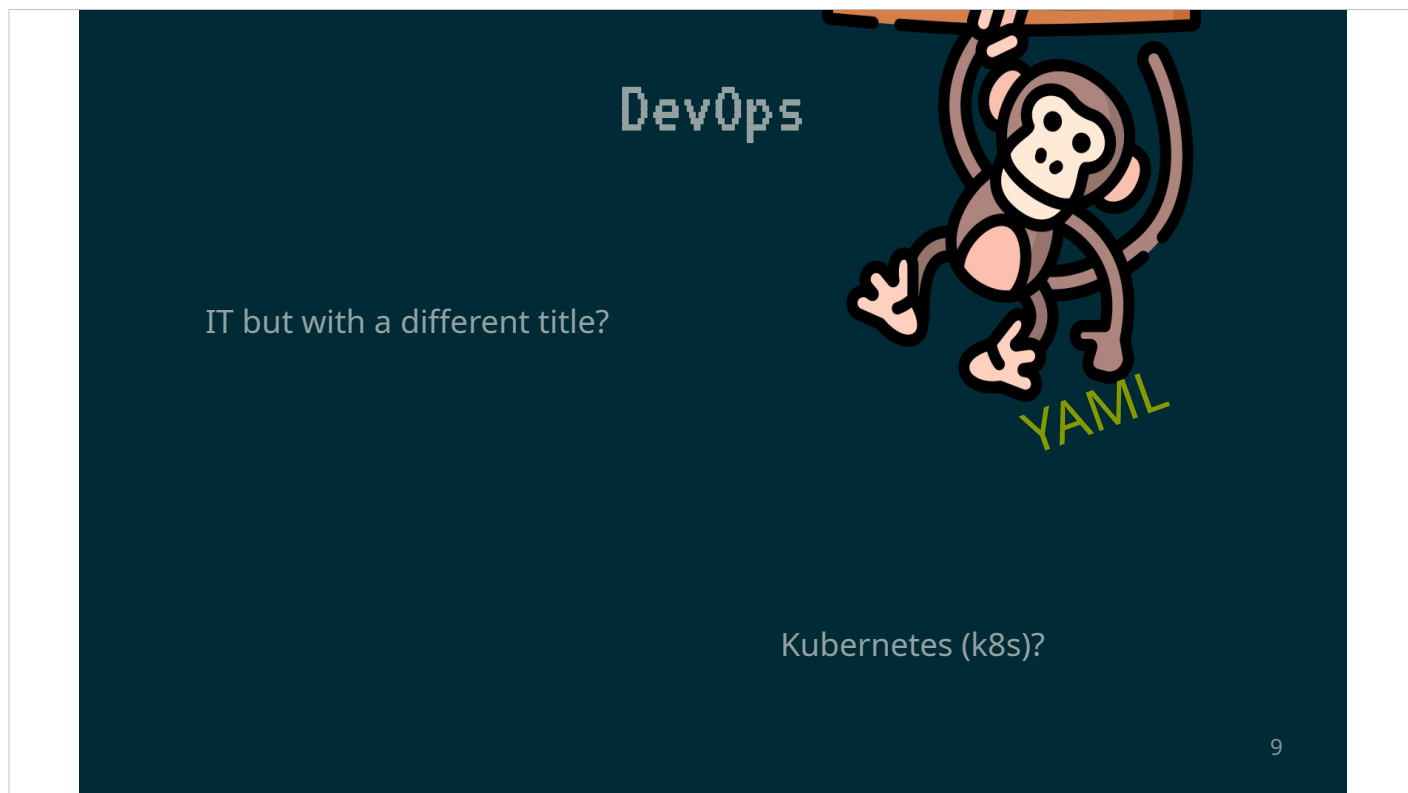
DevOps

IT but with a different title?

DevOps

IT but with a different title?

Kubernetes (k8s)?



[Monkey icons created by Freepik - Flaticon](https://www.flaticon.com/free-icons/monkey "monkey icons")

Ask a programmer to review
ten lines of code, [they'll] find
ten issues.

Ask a programmer to review ten lines of code, [they'll] find ten issues. Ask [them] to do five hundred lines, and [they] will say it looks good.

Ask a programmer to review ten lines of code, [they'll] find ten issues. Ask [them] to do five hundred lines, and [they] will say it looks good.

Gene Kim, "The DevOps Handbook"

Solving this problem:



14

1. quality
2. velocity
3. efficiency
4. we can choose 3!

Solving this problem:

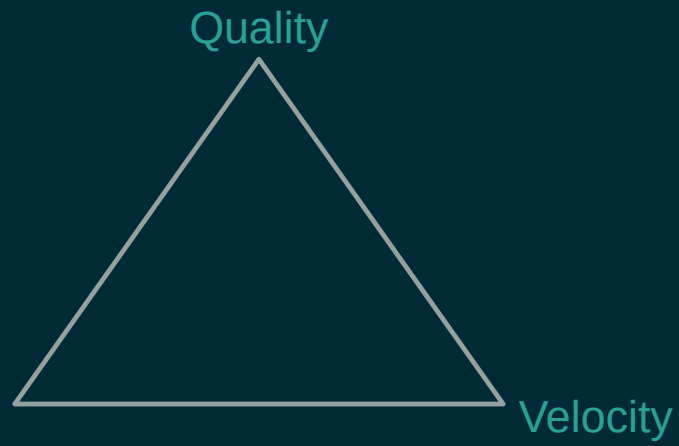
Quality



15

1. quality
2. velocity
3. efficiency
4. we can choose 3!

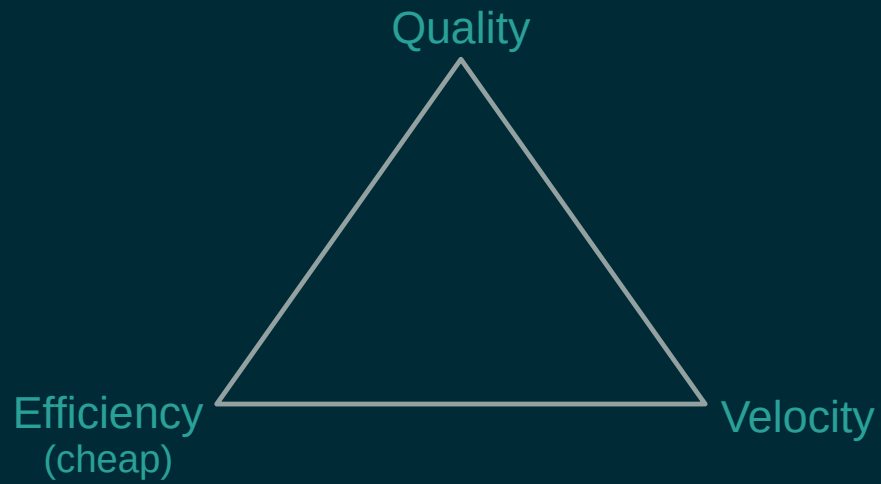
Solving this problem:



16

1. quality
2. velocity
3. efficiency
4. we can choose 3!

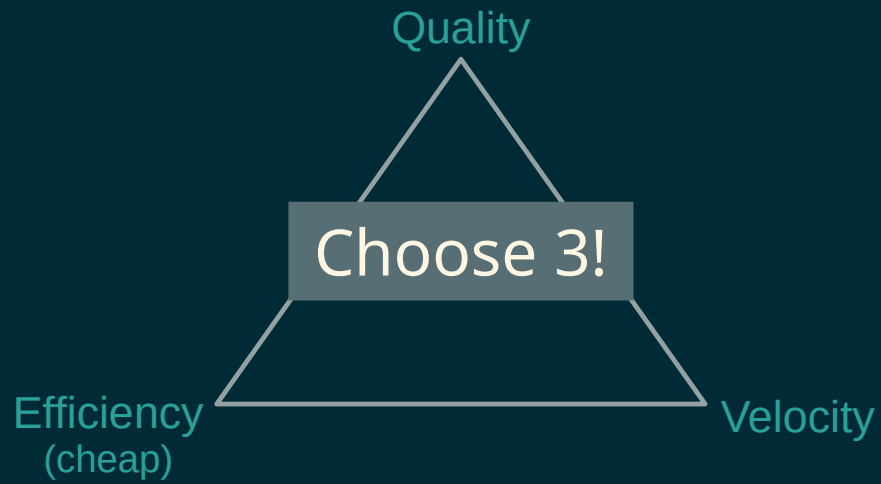
Solving this problem:



17

1. quality
2. velocity
3. efficiency
4. we can choose 3!

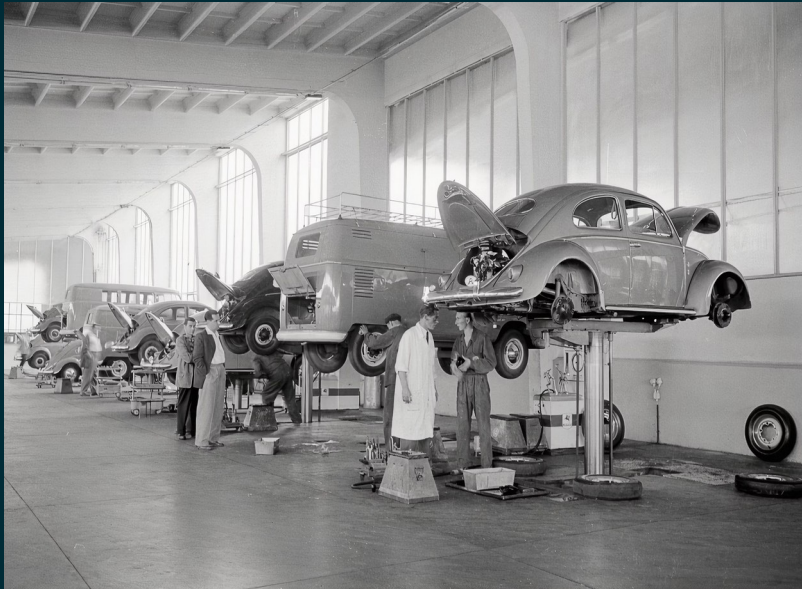
Solving this problem:



18

1. quality
2. velocity
3. efficiency
4. we can choose 3!

Developers in Operations



19

1. developers HAVE to know how their code is used, and how it gets deployed
2. you can't bolt on delivery after building a product
3. you can't just "figure out ops later" just like you can't "do security later"
4. every developer needs to know
5. not expert tho

Photo by [Austrian National Library](https://unsplash.com/@austriannationallibrary?utm_content=creditCopyText&utm_medium=referral&utm_source=unsplash) on [Unsplash](https://unsplash.com/photos/gray-scale-photo-of-man-in-white-shirt-standing-beside-vintage-car-DZ0LXbrFpQM?utm_content=creditCopyText&utm_medium=referral&utm_source=unsplash)

How to DevOps



Elite



Low

20

1. ****step 1: read the first few pages of this (ideally the whole thing)****
2. how to move your team to elite?

DORA '23 takeaways



21

1. establish a healthy culture
2. build with users in mind
3. unlock software delivery performance with faster code reviews
4. Amplify technical capabilities with quality documentation
 1. (gasp my agile!)
5. Flexible infrastructure with cloud predicts success
6. balance
7. fair work
 1. burnout is real and damages performance
8. if you have any responsibility for delivering software or managing those that do, you really owe it to yourself to read this
9. you may not agree with everything, but they have a LOT of evidence to back up their findings - even if they focus a bit too much on reverting problems rather than not pushing them to prod instead

DORA '23 takeaways



- culture

22

1. establish a healthy culture
2. build with users in mind
3. unlock software delivery performance with faster code reviews
4. Amplify technical capabilities with quality documentation
 1. (gasp my agile!)
5. Flexible infrastructure with cloud predicts success
6. balance
7. fair work
 1. burnout is real and damages performance
8. if you have any responsibility for delivering software or managing those that do, you really owe it to yourself to read this
9. you may not agree with everything, but they have a LOT of evidence to back up their findings - even if they focus a bit too much on reverting problems rather than not pushing them to prod instead

DORA '23 takeaways



- culture
- users in mind

23

1. establish a healthy culture
2. build with users in mind
3. unlock software delivery performance with faster code reviews
4. Amplify technical capabilities with quality documentation
 1. (gasp my agile!)
5. Flexible infrastructure with cloud predicts success
6. balance
7. fair work
 1. burnout is real and damages performance
8. if you have any responsibility for delivering software or managing those that do, you really owe it to yourself to read this
9. you may not agree with everything, but they have a LOT of evidence to back up their findings - even if they focus a bit too much on reverting problems rather than not pushing them to prod instead

DORA '23 takeaways



- culture
- users in mind
- faster code reviews

24

1. establish a healthy culture
2. build with users in mind
3. unlock software delivery performance with faster code reviews
4. Amplify technical capabilities with quality documentation
 1. (gasp my agile!)
5. Flexible infrastructure with cloud predicts success
6. balance
7. fair work
 1. burnout is real and damages performance
8. if you have any responsibility for delivering software or managing those that do, you really owe it to yourself to read this
9. you may not agree with everything, but they have a LOT of evidence to back up their findings - even if they focus a bit too much on reverting problems rather than not pushing them to prod instead

DORA '23 takeaways



- culture
- users in mind
- faster code reviews
- quality documentation → amplifies tech. ability

25

1. establish a healthy culture
2. build with users in mind
3. unlock software delivery performance with faster code reviews
4. Amplify technical capabilities with quality documentation
 1. (gasp my agile!)
5. Flexible infrastructure with cloud predicts success
6. balance
7. fair work
 1. burnout is real and damages performance
8. if you have any responsibility for delivering software or managing those that do, you really owe it to yourself to read this
9. you may not agree with everything, but they have a LOT of evidence to back up their findings - even if they focus a bit too much on reverting problems rather than not pushing them to prod instead

DORA '23 takeaways



- culture
- users in mind
- faster code reviews
- quality documentation → amplifies tech. ability
- infrastructure flexibility (cloud)

26

1. establish a healthy culture
2. build with users in mind
3. unlock software delivery performance with faster code reviews
4. Amplify technical capabilities with quality documentation
 1. (gasp my agile!)
5. Flexible infrastructure with cloud predicts success
6. balance
7. fair work
 1. burnout is real and damages performance
8. if you have any responsibility for delivering software or managing those that do, you really owe it to yourself to read this
9. you may not agree with everything, but they have a LOT of evidence to back up their findings - even if they focus a bit too much on reverting problems rather than not pushing them to prod instead

DORA '23 takeaways



- culture
- users in mind
- faster code reviews
- quality documentation → amplifies tech. ability
- infrastructure flexibility (cloud)
- Balance delivery speed, operational performance, and user focus

27

1. establish a healthy culture
2. build with users in mind
3. unlock software delivery performance with faster code reviews
4. Amplify technical capabilities with quality documentation
 1. (gasp my agile!)
5. Flexible infrastructure with cloud predicts success
6. balance
7. fair work
 1. burnout is real and damages performance
8. if you have any responsibility for delivering software or managing those that do, you really owe it to yourself to read this
9. you may not agree with everything, but they have a LOT of evidence to back up their findings - even if they focus a bit too much on reverting problems rather than not pushing them to prod instead

DORA '23 takeaways



- culture
- users in mind
- faster code reviews
- quality documentation → amplifies tech. ability
- infrastructure flexibility (cloud)
- Balance delivery speed, operational performance, and user focus
- Distribute work fairly

28

1. establish a healthy culture
2. build with users in mind
3. unlock software delivery performance with faster code reviews
4. Amplify technical capabilities with quality documentation
 1. (gasp my agile!)
5. Flexible infrastructure with cloud predicts success
6. balance
7. fair work
 1. burnout is real and damages performance
8. if you have any responsibility for delivering software or managing those that do, you really owe it to yourself to read this
9. you may not agree with everything, but they have a LOT of evidence to back up their findings - even if they focus a bit too much on reverting problems rather than not pushing them to prod instead

DORA '23 takeaways



- culture
- users in mind
- **faster code reviews**
- quality documentation → amplifies tech. ability
- infrastructure flexibility (cloud)
- Balance delivery speed, operational performance, and user focus
- Distribute work fairly

1. I don't only want to pontificate...
2. rest of the presentation will be a bit more on the technical side

Review Speed

31

1. we also call our work trunk-based development
2. Google says 1 business day or less |
<https://google.github.io/eng-practices/review/reviewer/speed.html>
3. 10 minute build upper limit
4. this is a spicy take

Review Speed

- 1 day review - Google

32

1. we also call our work trunk-based development
2. Google says 1 business day or less |
<https://google.github.io/eng-practices/review/reviewer/speed.html>
3. 10 minute build upper limit
4. this is a spicy take

Review Speed

- 1 day review – Google
- 10 min build – Kent Beck “Extreme Programming Explained”

33

1. we also call our work trunk-based development
2. Google says 1 business day or less |
<https://google.github.io/eng-practices/review/reviewer/speed.html>
3. 10 minute build upper limit
4. this is a spicy take

Review Speed

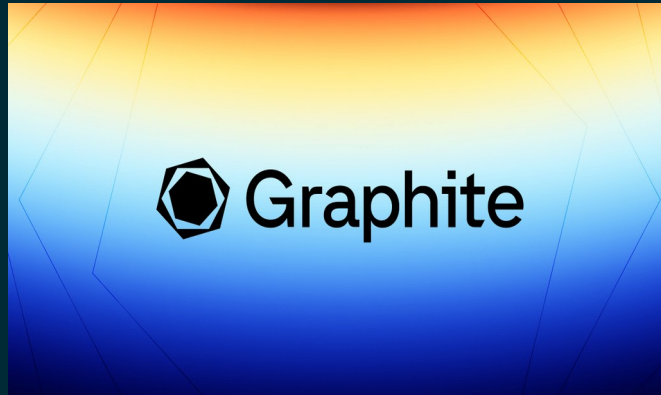
- 1 day review – Google
- 10 min build – Kent Beck “Extreme Programming Explained”



34

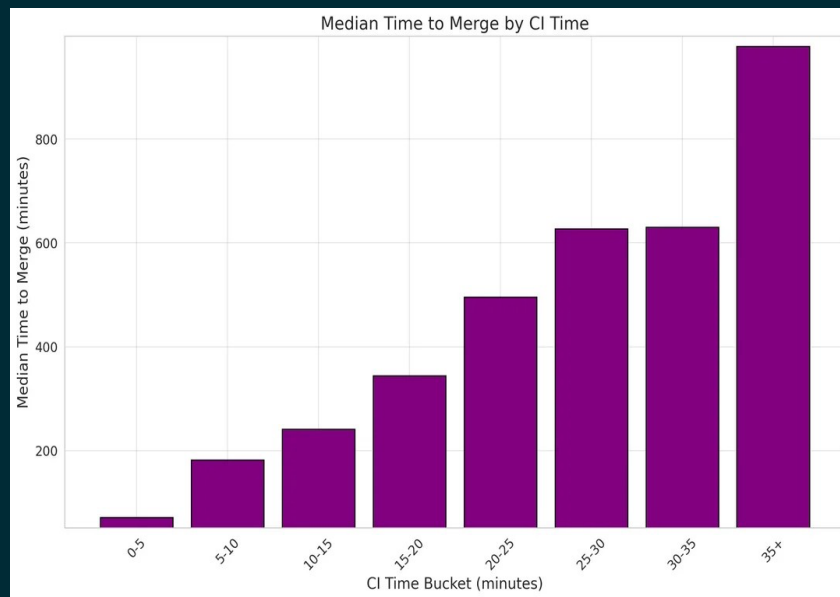
1. we also call our work trunk-based development
2. Google says 1 business day or less |
<https://google.github.io/eng-practices/review/reviewer/speed.html>
3. 10 minute build upper limit
4. this is a spicy take

“How long should your CI take?”



35

Blog post from graphite



graphite.dev

36

Their numbers

Our Numbers

REDACTED

37

1. not same metrics as github

Sneak peak

CI Speed



CI Speed

- No lower bound

CI Speed

- No lower bound
- 1-2 hour merge penalty / for every +5 min CI

CI Speed

- No lower bound
- 1-2 hour merge penalty / for every +5 min CI

`graphite.dev` - "How long should your CI take?"

Side-adventure: Our stats

42

Dive in

Side-adventure: Our stats



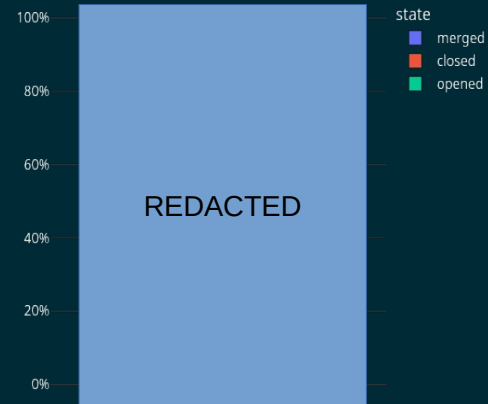
43

Dive in

Side-adventure: Our stats

*Only visible projects with > 25 MRs

MR Status



44

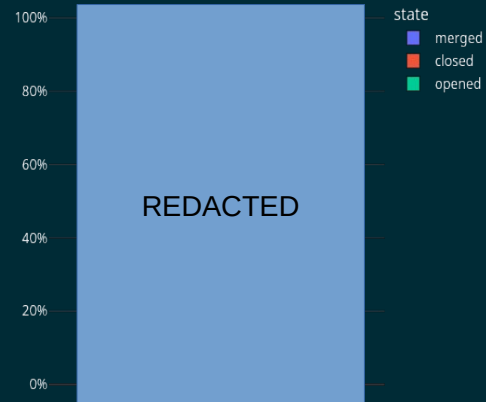
1. since 2022, with projects that have at least 25 mrs
 1. rule out hobby projects
2. shout out [REDACTED]

Side-adventure: Our stats

*Only visible projects with > 25 MRs

> 23k MRs

MR Status

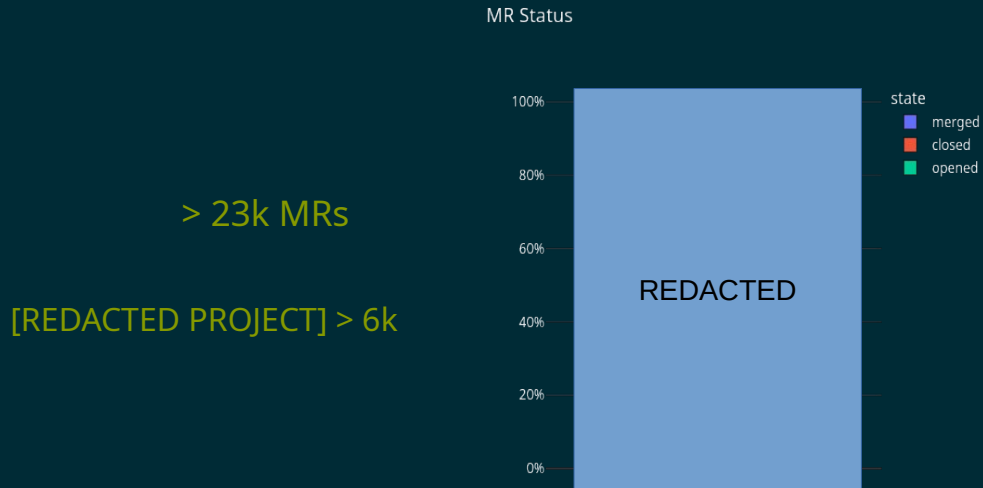


45

1. since 2022, with projects that have at least 25 mrs
 1. rule out hobby projects
2. shout out [REDACTED]

Side-adventure: Our stats

*Only visible projects with > 25 MRs



46

1. since 2022, with projects that have at least 25 mrs
 1. rule out hobby projects
2. shout out [REDACTED]

CI Jobs

- 58% of MRs: 1+ pipeline

47

1. this gets better the longer a project exists (60% on projs with >100 MRs)
2. don't queue for long
3. but can take a while, with a LONG tail
 1. partially due to hardware limitations (guessing, anecdotal)
 2. instability cannot be solely blamed on CI

CI Jobs

- 58% of MRs: 1+ pipeline
- XX (mean) XX (median) seconds queuing time

48

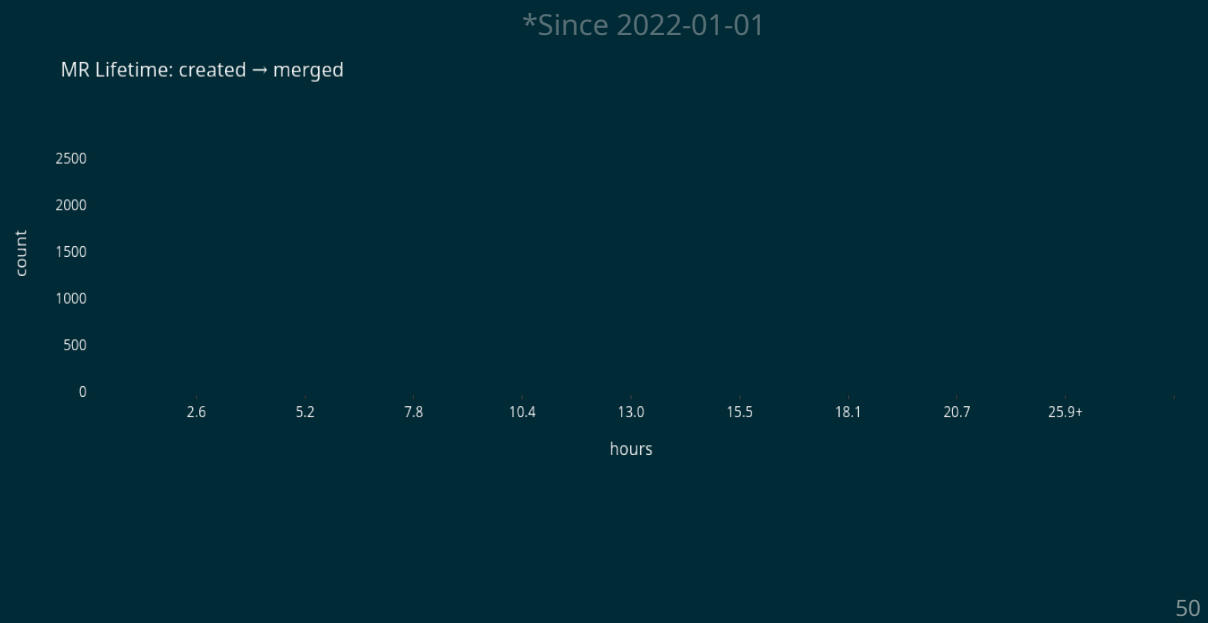
1. this gets better the longer a project exists (60% on projs with >100 MRs)
2. don't queue for long
3. but can take a while, with a LONG tail
 1. partially due to hardware limitations (queueing, anecdotal)
 2. instability cannot be solely blamed on CI

CI Jobs

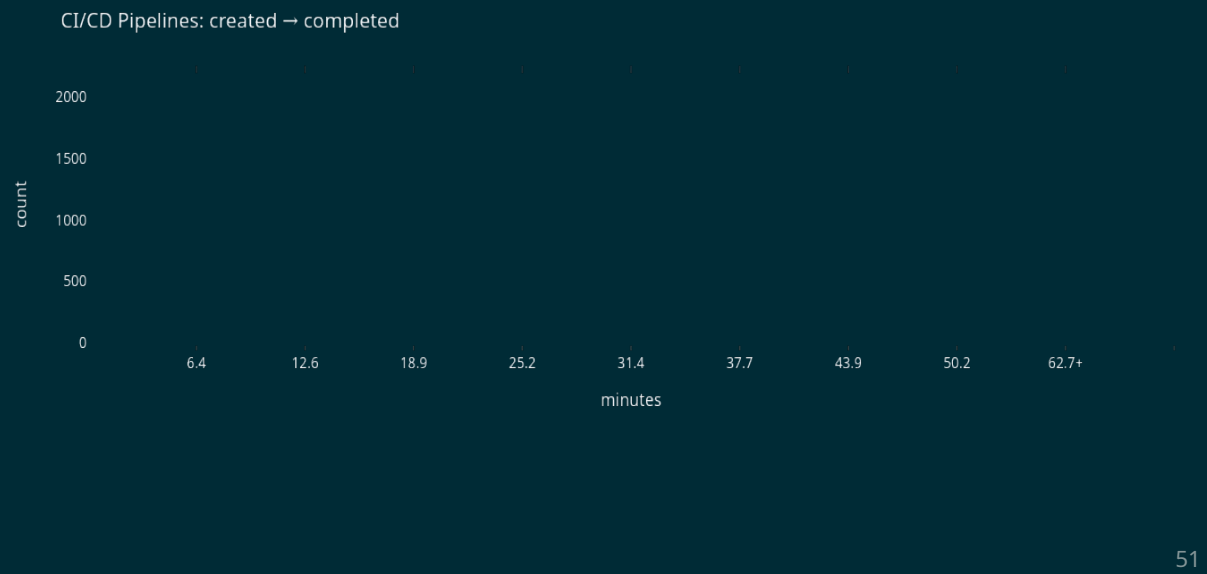
- 58% of MRs: 1+ pipeline
- XX (mean) XX (median) seconds queuing time
- XX (mean) X.X (median) minutes run time (excl. draft MRs)

49

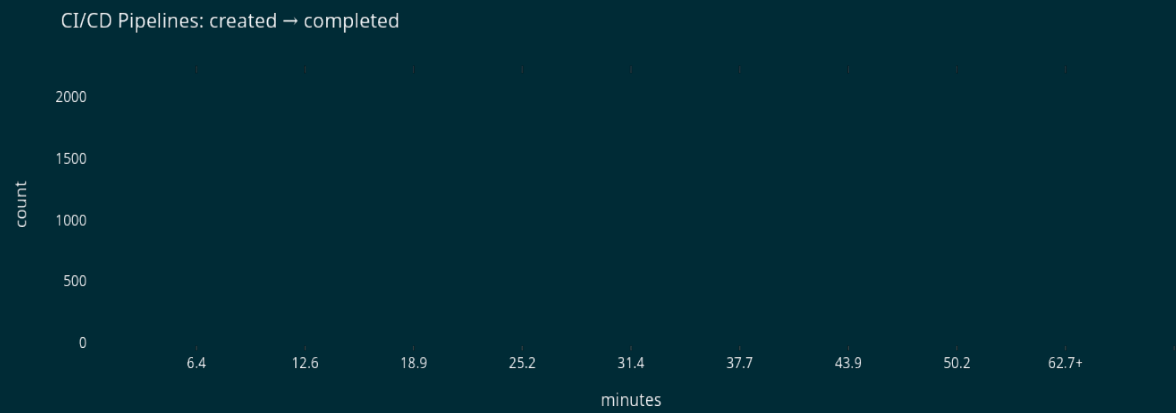
1. this gets better the longer a project exists (60% on projs with >100 MRs)
2. don't queue for long
3. but can take a while, with a LONG tail
 1. partially due to hardware limitations (queueing, anecdotal)
 2. instability cannot be solely blamed on CI



1. longest was 272 days



1. a nice distribution
2. I didn't discriminate vs protected (main) branches, where a longer pipeline is more acceptable (after merged!)



[REDACTED]/hgibson/ci-stats

52

1. a nice distribution
2. I didn't discriminate vs protected (main) branches, where a longer pipeline is more acceptable (after merged!)

SO WHAT

Actionable ideas

54

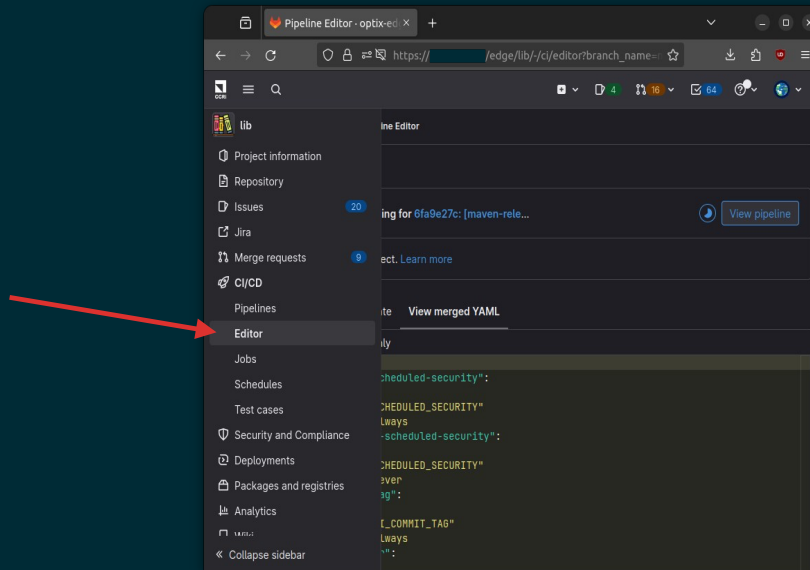
SO WHAT

The Merged YAML View

55

1. see what gitlab sees

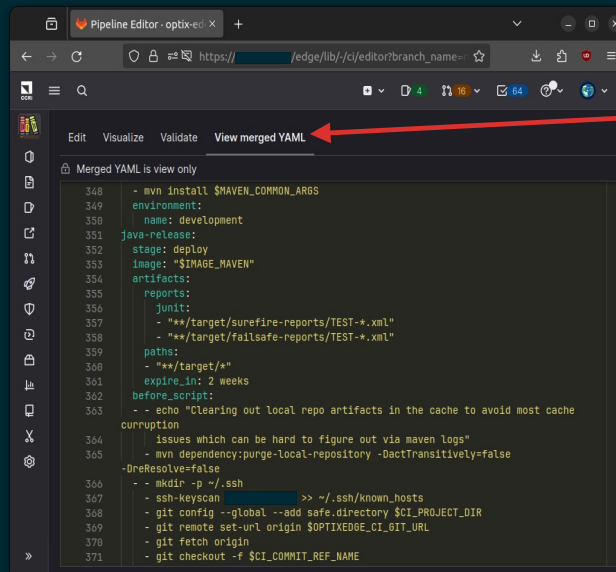
The Merged YAML View



56

1. see what gitlab sees

The Merged YAML View



```
348 - mvn install $MAVEN_COMMON_ARGS
349
350 environment:
351   name: development
352 java-release:
353   stage: deploy
354   image: "IMAGE_MAVEN"
355   artifacts:
356     reports:
357       junit:
358         - "**/target/surefire-reports/TEST-*.xml"
359         - "**/target/failsafe-reports/TEST-*.xml"
360     paths:
361       - "**/target/*"
362     expire_in: 2 weeks
363   before_script:
364     - - echo "Clearing out local repo artifacts in the cache to avoid most cache
365       corruption
366         issues which can be hard to figure out via maven logs"
367       - mvn dependency:purge-local-repository -DactTransitively=false
368   -DreResolve=false
369   - mkdir -p ~/.ssh
370   - ssh-keyscan >> ~/.ssh/known_hosts
371   - git config --global --add safe.directory $CI_PROJECT_DIR
372   - git remote set-url origin $OPTIXEDGE_CI_GIT_URL
373   - git fetch origin
374   - git checkout -f $CI_COMMIT_REF_NAME
```

57

1. see what gitlab sees

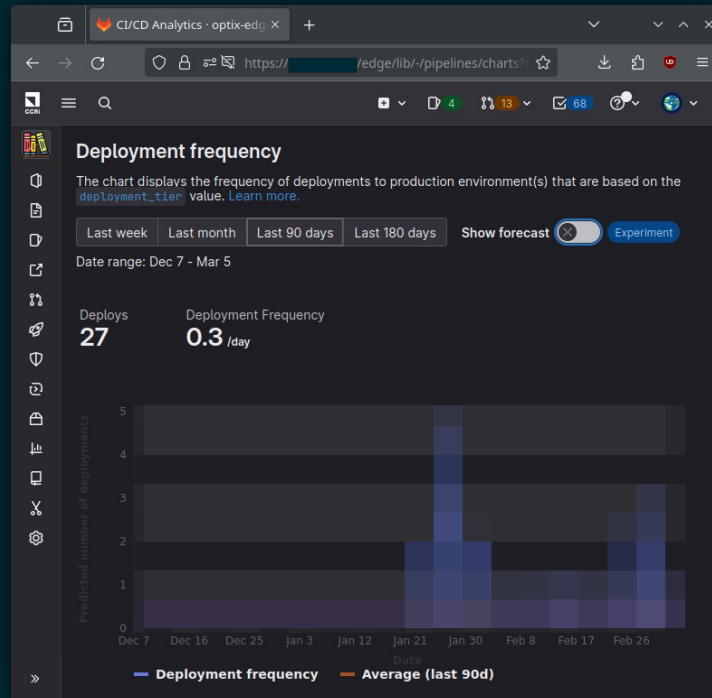
Getting changes into production



```
deploy_to_prod:  
  script: ...  
  environment:  
    name: production
```

Free stats!!

Env to “production”



60

Example of env production



Provide flexibility
not solutions

61

Mission command

Give tools not answers

```
include:
  - project: "edge/common-cicd"
    ref: "0.13.1"
    file: "gitlab-ci/job-templates/release.yml"
gitlab-release:
  extends: .gitlab_release_job
  stage: deploy
```

```
include:
  - project: 'ccri/ccri-ci-templates'
    ref: '2.4.1'
    file: 'maven/default-maven-build.yml'
variables:
  CI_BUILD: 'false'
  CI_PAGES: 'false'
  CI_GEMNASIUM: 'false'
  CI_CONTAINER_SCAN: 'false'
  CI_LICENSE_SCAN: 'false'
  CI_METRICS: 'false'
  CI_QUALITY: 'false'
```

62

We redesigned our CI

Corporate CI == mess if not standard workflow

Adding another job

> make sure all builds compile

```

include:
  - project: "edge/common-cicd"
    ref: "0.13.1"
    file: "gitlab-ci/job-templates/release.yml"
  - project: "edge/common-cicd"
    ref: "0.13.1"
    file: "gitlab-ci/job-templates/maven.yml"
  - project: "edge/common-cicd"
    ref: "0.13.1"
    file: "gitlab-ci/maven-settings.yml"
maven-build:
  extends: .maven_job
  stage: build
  script:
    - mvn clean compile -DmyParam $MAVEN_COMMON_ARGS
gitlab-release:
  extends: .gitlab_release_job
  stage: deploy

```

```

include:
  - project: 'ccri/ccri-ci-templates'
    ref: '2.4.1'
    file: 'maven/default-maven-build.yml'
variables:
  CI_BUILD: 'true'
  CI_PAGES: 'false'
  CI_GEMNASIUM: 'false'
  CI_CONTAINER_SCAN: 'false'
  CI_LICENSE_SCAN: 'false'
  CI_METRICS: 'false'
  CI_QUALITY: 'false'
  SAST_DISABLED: 'true'
  MAVEN_EXTRA_MAIN_ARGS: '-DmyParam'
main_build:
  variables:
    MAVEN_ARGS: 'clean compile'
  before_script:
    # overwrite the before_script
    - 'true'

```

35. add a few imports and declare job vs "knowing" which params to check

1. don't read into the length
2. I ran both of these pipelines, the left took me a lot less time to debug and add new job

```

include:
  - project: "edge/common-cicd"
    ref: "0.13.1"
    file: "gitlab-ci/job-templates/release.yml"
  - project: "edge/common-cicd"
    ref: "0.13.1"
    file: "gitlab-ci/job-templates/maven.yml"
  - project: "edge/common-cicd"
    ref: "0.13.1"
    file: "gitlab-ci/maven-settings.yml"
maven-build:
  extends: .maven_job
  stage: build
  script:
    - mvn clean compile -DmyParam $MAVEN_COMMON_ARGS
gitlab-release:
  extends: .gitlab_release_job
  stage: deploy

```

```

include:
  - project: 'ccri/ccri-ci-templates'
    ref: '2.4.1'
    file: 'maven/default-maven-build.yml'
variables:
  CI_BUILD: 'true'
  CI_PAGES: 'false'
  CI_GEMNASIUM: 'false'
  CI_CONTAINER_SCAN: 'false'
  CI_LICENSE_SCAN: 'false'
  CI_METRICS: 'false'
  CI_QUALITY: 'false'
  SAST_DISABLED: 'true'
  MAVEN_EXTRA_MAIN_ARGS: '-DmyParam'
main_build:
  variables:
    MAVEN_ARGS: 'clean compile'
  before_script:
    # overwrite the before_script
    - 'true'

```

35. add a few imports and declare job vs "knowing" which params to check

1. don't read into the length
2. I ran both of these pipelines, the left took me a lot less time to debug and add new job

Share templates not jobs

66

36. here's the `.maven_job` I used before

1. ACTION: provide templates
2. ACTION: don't prescribe `script:` in job templates
3. By providing a template that doesn't automatically run, you make it an "opt in" rather than "opt out"

Share templates not jobs

opt-in over opt-out

```
.maven_job: &maven_job
  stage: main
  image: $IMAGE_MAVEN
  artifacts:
    reports:
      junit:
        - "**/target/surefire-reports/TEST-*.xml"
        - "**/target/failsafe-reports/TEST-*.xml"
  paths:
    - "**/target/*"
  expire_in: 2 weeks
  before_script:
    - 'echo "Clearing out local repo artifacts in the cache to avoid most cache corruption issues which can be hard to figure out via maven logs"'
    - 'mvn dependency:purge-local-repository -DactTransitively=false -DreResolve=false'
```

36. here's the `.maven\_job` I used before

1. ACTION: provide templates
2. ACTION: don't prescribe `script:` in job templates
3. By providing a template that doesn't automatically run, you make it an "opt in" rather than "opt

out"

Reliability unlocks performance

68

37. Now for managers and tech leads

1. give consistency where you can

Reliability unlocks performance

consistency

69

37. Now for managers and tech leads

1. give consistency where you can

Review Speed

70

38. Examples that improve review speed

1. what is a reviewer looking at?
 1. style? no
 2. correctness? not really
 3. better ideas? no
2. These things matter for the veteran engineer just as much as the brand new engineer
3. reduce "friction" to delivering value to the customer!

Review Speed

- Enforced style

71

38. Examples that improve review speed

1. what is a reviewer looking at?
 1. style? no
 2. correctness? not really
 3. better ideas? no
2. These things matter for the veteran engineer just as much as the brand new engineer
3. reduce "friction" to delivering value to the customer!

Review Speed

- Enforced style
- Tests = Correctness

72

38. Examples that improve review speed

1. what is a reviewer looking at?
 1. style? no
 2. correctness? not really
 3. better ideas? no
2. These things matter for the veteran engineer just as much as the brand new engineer
3. reduce "friction" to delivering value to the customer!

Review Speed

- Enforced style
- Tests = Correctness
- Clear review guidelines

73

38. Examples that improve review speed

1. what is a reviewer looking at?
 1. style? no
 2. correctness? not really
 3. better ideas? no
2. These things matter for the veteran engineer just as much as the brand new engineer
3. reduce "friction" to delivering value to the customer!

Review Speed

- Enforced style
- Tests = Correctness
- Clear review guidelines
- Clear ... everything

74

38. Examples that improve review speed

1. what is a reviewer looking at?
 1. style? no
 2. correctness? not really
 3. better ideas? no
2. These things matter for the veteran engineer just as much as the brand new engineer
3. reduce "friction" to delivering value to the customer!

Review Speed

- Enforced style
- Tests = Correctness
- Clear review guidelines
- Clear ... everything
 - Who approves?

75

38. Examples that improve review speed

1. what is a reviewer looking at?
 1. style? no
 2. correctness? not really
 3. better ideas? no
2. These things matter for the veteran engineer just as much as the brand new engineer
3. reduce "friction" to delivering value to the customer!

Review Speed

- Enforced style
- Tests = Correctness
- Clear review guidelines
- Clear ... everything
 - Who approves?
 - What am I allowed to commit?

76

38. Examples that improve review speed

1. what is a reviewer looking at?
 1. style? no
 2. correctness? not really
 3. better ideas? no
2. These things matter for the veteran engineer just as much as the brand new engineer
3. reduce "friction" to delivering value to the customer!

Review Speed

- Enforced style
- Tests = Correctness
- Clear review guidelines
- Clear ... everything
 - Who approves?
 - What am I allowed to commit?
 - What changes do you want?

77

38. Examples that improve review speed

1. what is a reviewer looking at?
 1. style? no
 2. correctness? not really
 3. better ideas? no
2. These things matter for the veteran engineer just as much as the brand new engineer
3. reduce "friction" to delivering value to the customer!

Review Speed

- Enforced style
- Tests = Correctness
- Clear review guidelines
- Clear ... everything
 - Who approves?
 - What am I allowed to commit?
 - What changes do you want?
 - ...

78

38. Examples that improve review speed

1. what is a reviewer looking at?
 1. style? no
 2. correctness? not really
 3. better ideas? no
2. These things matter for the veteran engineer just as much as the brand new engineer
3. reduce "friction" to delivering value to the customer!

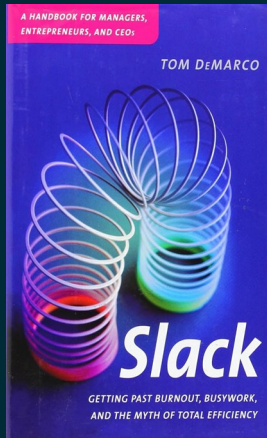
References

79

39. ... references

1. Slack = skim it, takes a saturday afternoon - engineers are not fungible
2. DevOps handbook - for engineers AND managers
3. DORA -
 1. this is what matters RIGHT NOW,
 2. managers: at least read the first few pages you are doing yourself a disservice
 3. very few things actually configured with YAML (i.e. team culture)

References



80

39. ... references

1. Slack = skim it, takes a saturday afternoon - engineers are not fungible
2. DevOps handbook - for engineers AND managers
3. DORA -
 1. this is what matters RIGHT NOW,
 2. managers: at least read the first few pages you are doing yourself a disservice
 3. very few things actually configured with YAML (i.e. team culture)

References

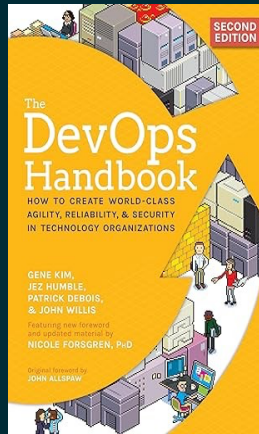


81

39. ... references

1. Slack = skim it, takes a saturday afternoon - engineers are not fungible
2. DevOps handbook - for engineers AND managers
3. DORA -
 1. this is what matters RIGHT NOW,
 2. managers: at least read the first few pages you are doing yourself a disservice
 3. very few things actually configured with YAML (i.e. team culture)

References

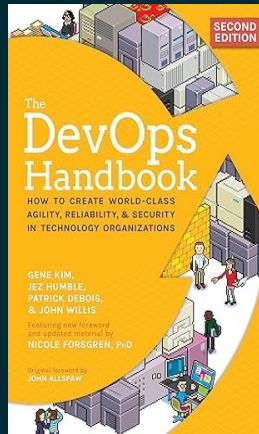


82

39. ... references

1. Slack = skim it, takes a saturday afternoon - engineers are not fungible
2. DevOps handbook - for engineers AND managers
3. DORA -
 1. this is what matters RIGHT NOW,
 2. managers: at least read the first few pages you are doing yourself a disservice ignoring the exec. summary
 3. very few things actually configured with YAML (i.e. team culture)

References

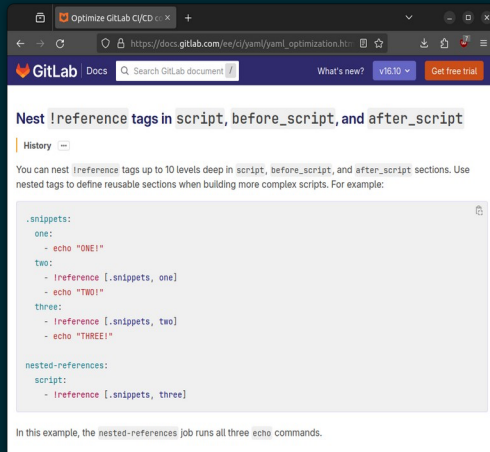


83

39. ... references

1. Slack = skim it, takes a saturday afternoon - engineers are not fungible
2. DevOps handbook - for engineers AND managers
3. DORA -
 1. this is what matters RIGHT NOW,
 2. managers: at least read the first few pages you are doing yourself a disservice ignoring the exec. summary
 3. very few things actually configured with YAML (i.e. team culture)

Cont.



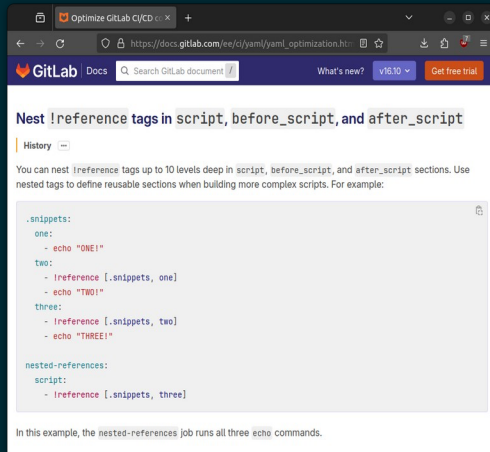
docs.gitlab.com

84

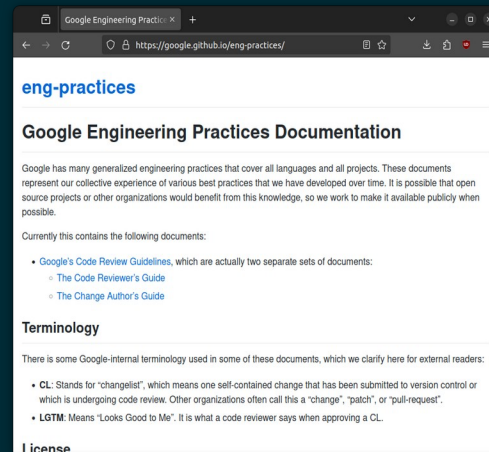
40. references cont.

1. GitLab-CI is complicated and does weird things, but I learned everything from the docs and you can too
2. google/meta/IBM spent a bunch of money to research write things, may as well read them

Cont.



docs.gitlab.com/ee/ci/yaml/yaml_optimization.html

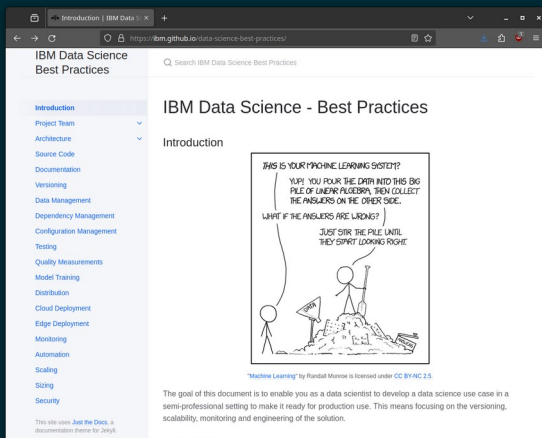


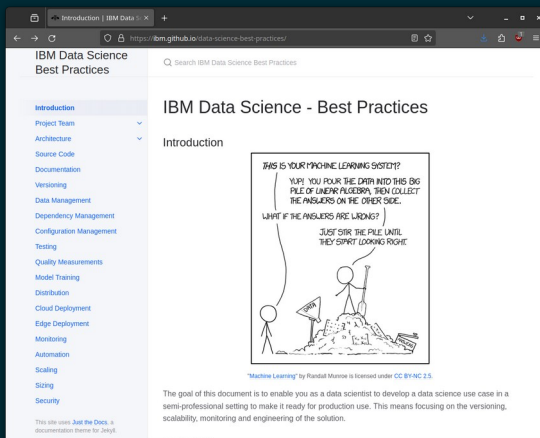
Google eng-practices

85

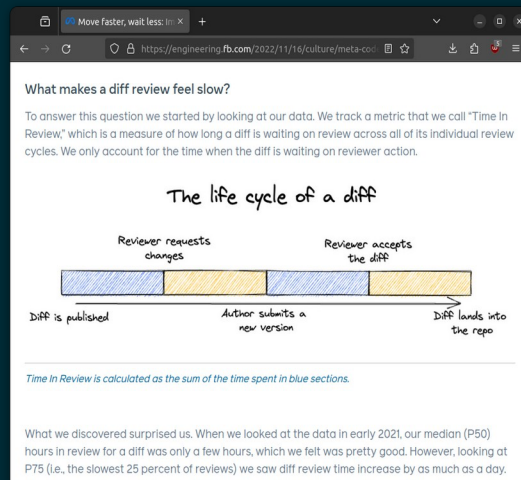
40. references cont.

1. GitLab-CI is complicated and does weird things, but I learned everything from the docs and you can too
2. google/meta/IBM spent a bunch of money to research write things, may as well read them





IBM GitHub.io



Meta Engineering Blog

87

Thanks!

Questions?