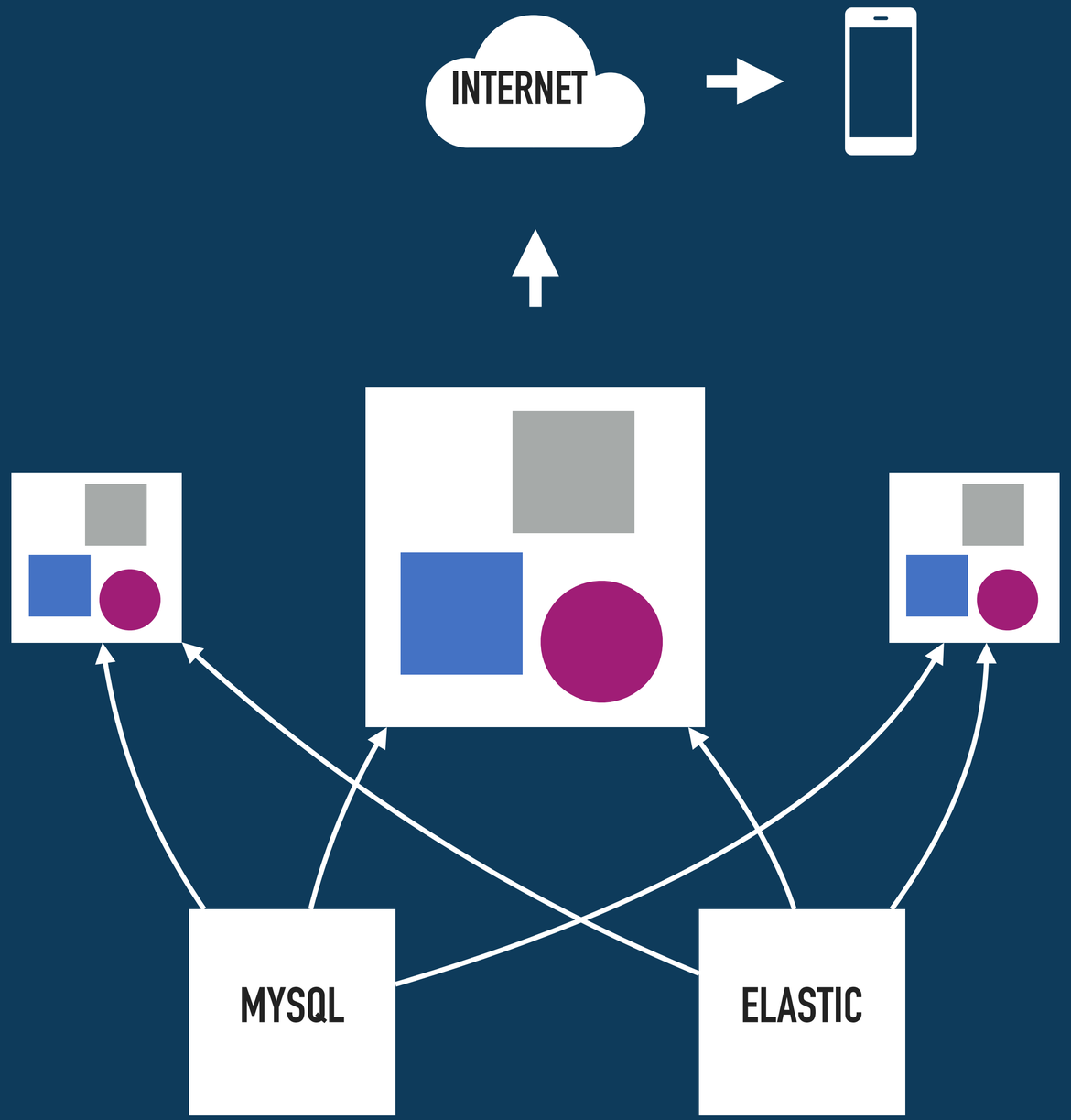
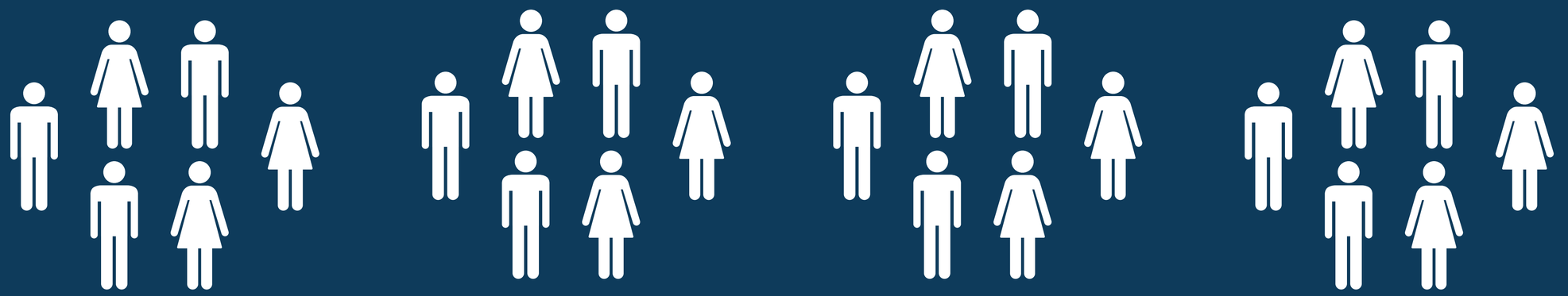
REAL WORLD CD
LEARN, ADAPT & IMPROVE

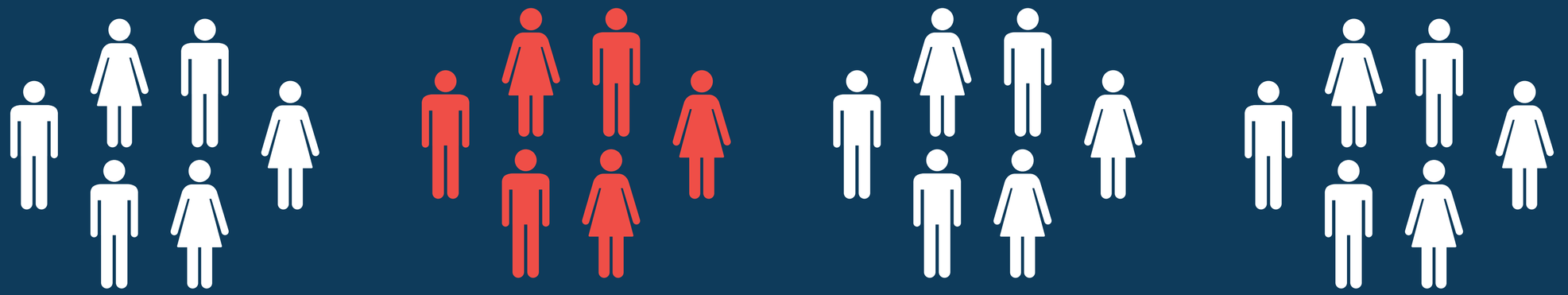
MICHIEL ROOK
@MICHIELTCS


LARGE SEMI-GOV
IN NL


INTERNET
MYSQL
ELASTIC



TOPIC TEAMS



FIRE TEAM

PACKAGE/DEP UPDATES

BUG TRIAGE

RELEASE ROLLOUT

A TYPICAL RELEASE...

Releasing a new version

Creating a new release is done at sprint end. A new release contains all the work from a sprint. As such, the *release-candidate* is based on the *develop* branch.

Make sure your local develop branch is up-to-date!

```
git checkout develop
git checkout -b release-candidate
git push
```

Deploy to Acceptance

- Create an artefact of the application on master by running the `mvp build master` job with parameter VERSION set to `2.0` and BRANCH_SPECIFIER to `release-candidate`. (this will start the `mvpacc deploy` job in Jenkins automatically)

Let the product owner or tester accept the change and verify that the acc environment is in the exact state they want prod to be. Any required changes should be as small as possible, and applied to the *release branch*. After the final OK, the release branch can be merged into *master*. Do this together with someone else if possible.

Make sure your local master branche is up-to-date!

```
git checkout master
git pull
git merge release-candidate
```

- Tag a new release with a new tag number, using the minor or major level Example: 1.18.1 becomes 1.19.0, etc.

```
git tag [tag number]
git push origin --tags
```

If there where commits on the *release-candidate* branch, they should also be applied to develop. Otherwise, solved problems will reappear later. Do this together with someone else if possible.

Make sure your local develop branche is up-to-date!

```
git checkout develop
git merge release-candidate
```

The release-candidate branch can now be deleted.

Hotfixing

A hotfix is a fix for something already in production. As such, your fix should be based on the *master* branch.

Make sure your local master branch is up-to-date!

```
git checkout master
git checkout -b hotfix/fix-some-crucial-bug
```

Apply your fix to this new branch and commit.

```
git push
```

You can create a pull request for peer review, but do not merge it. It should be checked on acceptance first.

Deploy to Acceptance

- Create an artefact of the application on master by running the `mvp build master` job with parameter VERSION set to `2.0` and BRANCH_SPECIFIER to `hotfix/fix-some-crucial-bug`. (this will start the `mvpacc deploy` job in Jenkins automatically)

Let the product owner or tester accept the change and verify that the acc environment is in the exact state they want prod to be. Assuming all is well, you can now merge the hotfix manually into master. Do this together with someone else if possible.

```
git checkout master
git merge hotfix/fix-some-crucial-bug
```

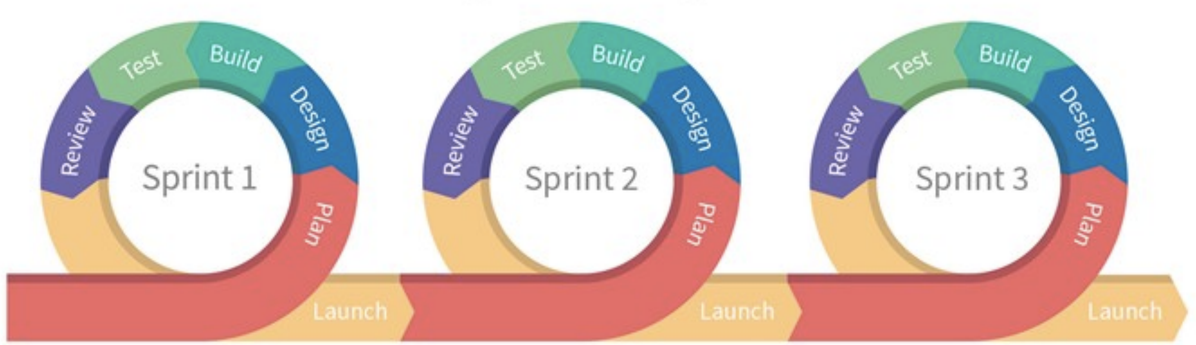
- Tag a new release with a new tag number, using the patch level Example: 1.18.1 becomes 1.18.2, etc.

```
git tag [tag number]
git push origin --tags
```

Every hotfix should also be merged into develop. Otherwise, solved problems will reappear later. Do this together with someone else if possible.

RELEASE CHECKLIST

@michieltcs



every
2 weeks

Create release

Thursday
(sprint end)

Test release

Monday
(sprint start)

Deploy release

Tuesday

1. BRANCH RC
2. DEPLOY TO ACC
3. SIGNAL TESTER(S)
4. SIGNAL PO
5. FIX ON BRANCH
6. GET APPROVAL
7. DEPLOY TO PROD
8. VERIFY PROD

2-3 DAYS MANUAL WORK

... HOT FIXES?

GOALS

REDUCE COST

FAST FEEDBACK

REDUCE TOIL

REDUCE TOIL

the kind of work tied to running a production service that tends to be manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows

PUT PO/BIZ BACK IN DRIVER'S SEAT

WHY CD THEN?

MAKE THINGS SMALL

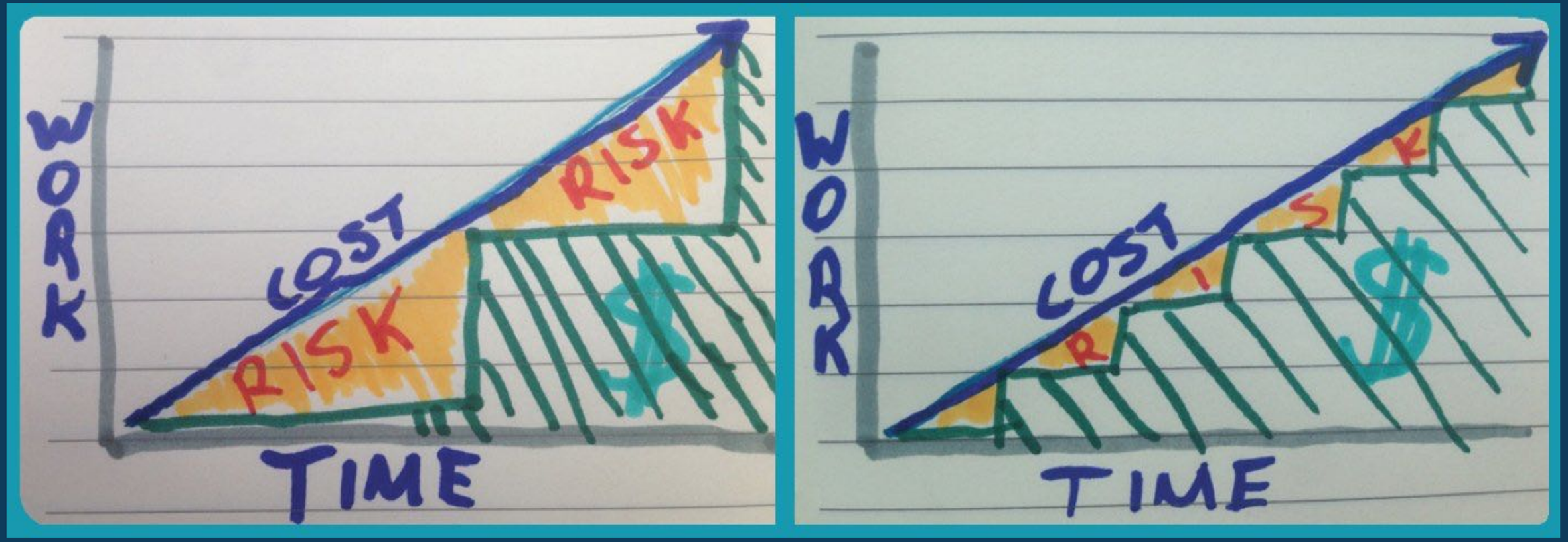
BIG STEPS

FAIL BIG

SMALL STEPS

FAIL SMALL

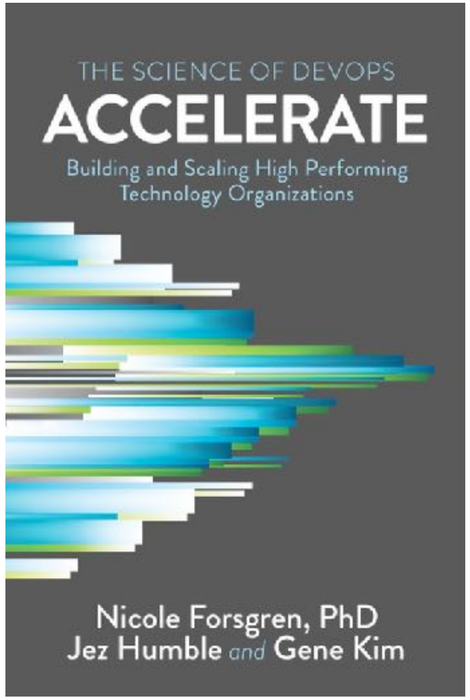
IF IT HURTS
DO IT MORE OFTEN



$ = REALIZED VALUE

CREDITS TO @FGOULDING

HIGH VS. LOW PERFORMING TEAMS



IT performance metrics	2017
Deployment frequency	46x more frequent
Lead time for changes	440x faster
Mean time to recover (MTTR)	96x faster
Change failure rate	5x lower (1/5 as likely)


APPROACH

3 PHASES

DEFER SPLITTING THE MONOLITH

INCREASE TEMPO & AUTOMATION

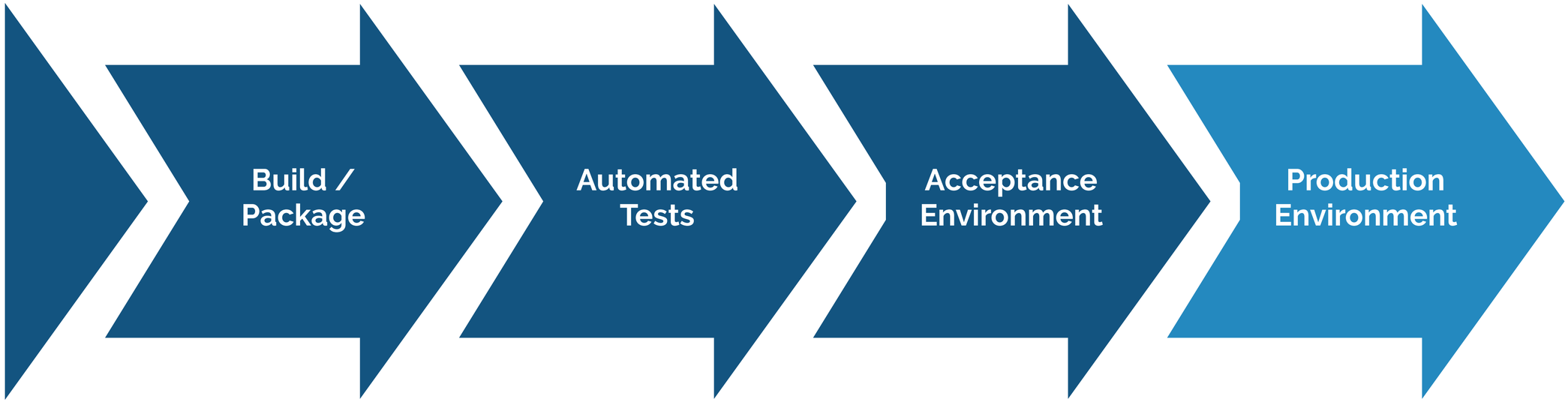
SINGLE PIPELINE

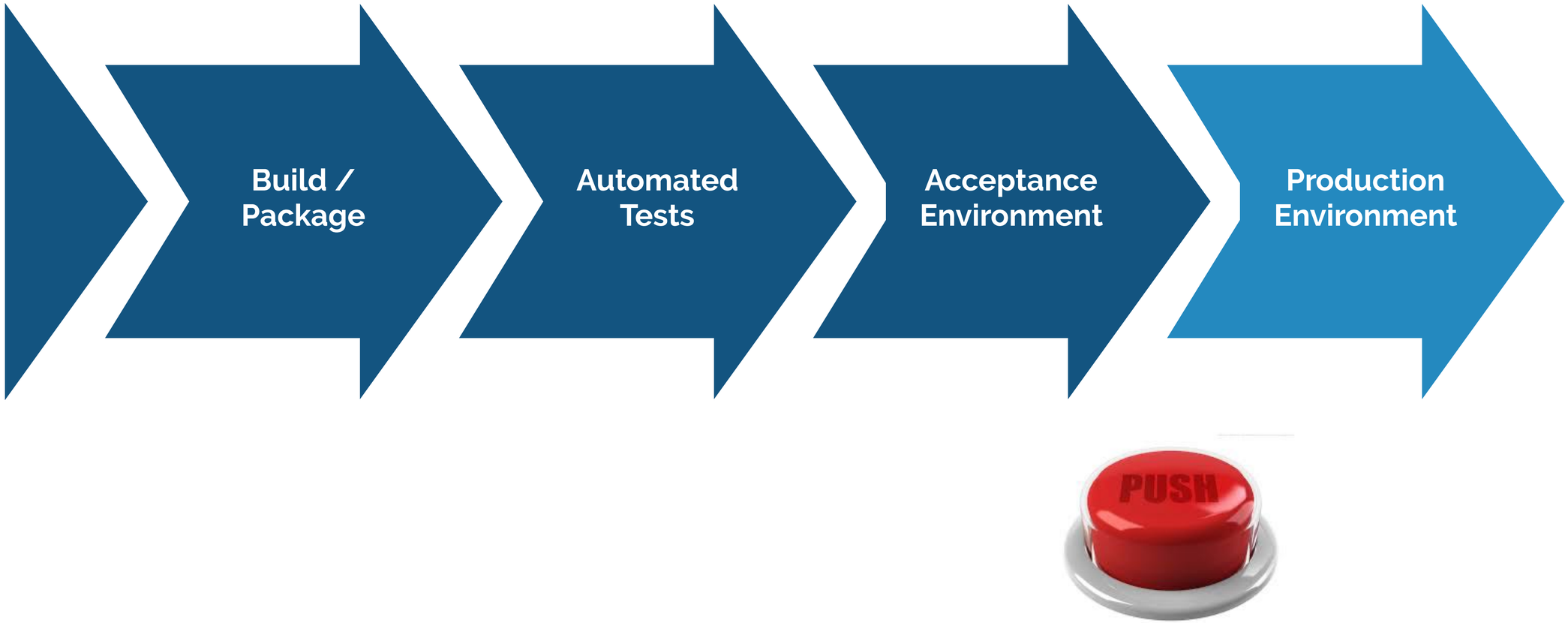
IMPROVEMENTS & TESTING

PHASE 1: INCREASING RELEASE CADENCE

WEEKLY RELEASE

AUTOMATION


Build /
Package
Automated
Tests
Acceptance
Environment
Production
Environment


Build /
Package
Automated
Tests
Acceptance
Environment
Production
Environment
PUSH

PIPELINE AS CODE

@michieltcs

jenkins [redacted] #696 ×

Michiel

Secure | https://jenkins.[redacted]

[redacted] 696

Pipeline | Changes | Tests | Artifacts | Login

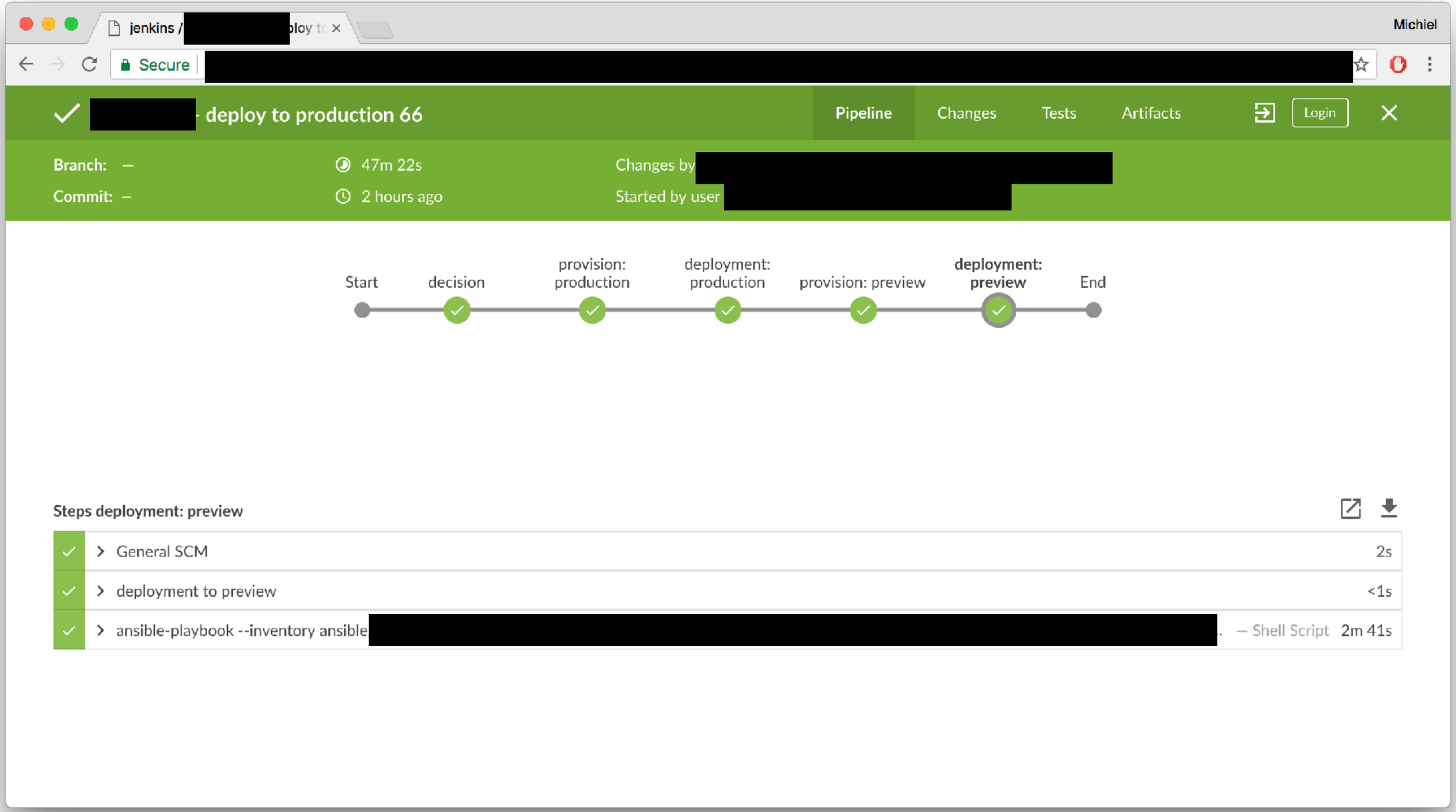
Branch: –
Commit: –

20m 3s
12 minutes ago

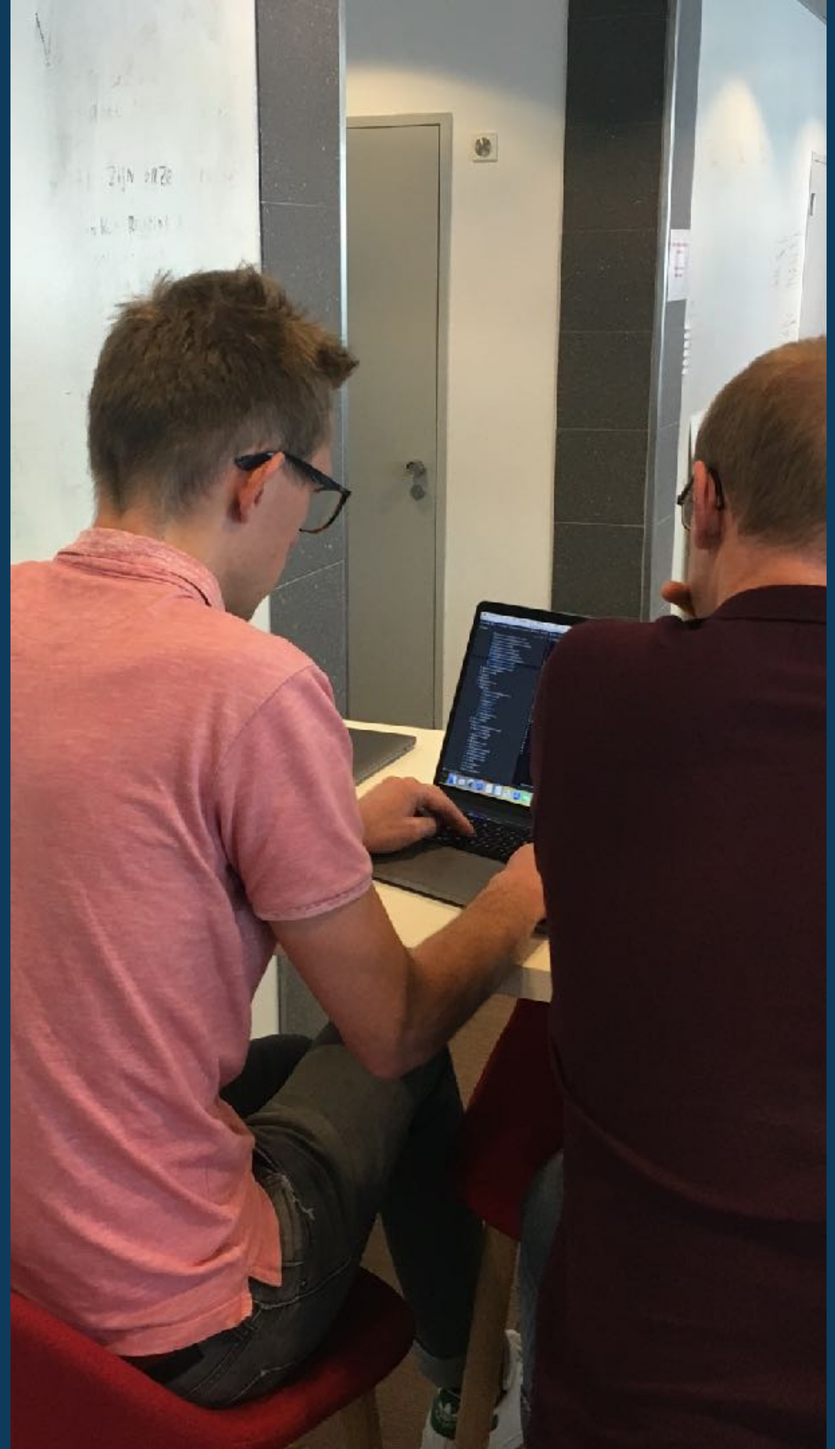
Changes by [redacted]
Started by an SCM change

Steps trigger

✓	Building [redacted]ploy to production	<1s
✓	nightwatch/report/cucumber_junit.xml — Archive JUnit-formatted test results	<1s


jenkins /
Michiel
Secure
deploy to production 66
Pipeline
Changes
Tests
Artifacts
Login
Branch: –
Commit: –
47m 22s
2 hours ago
Changes by
Started by user
Start
decision
provision: production
deployment: production
provision: preview
deployment: preview
End
Steps deployment: preview
General SCM 2s
deployment to preview <1s
ansible-playbook --inventory ansible
— Shell Script 2m 41s

WAY OF WORKING



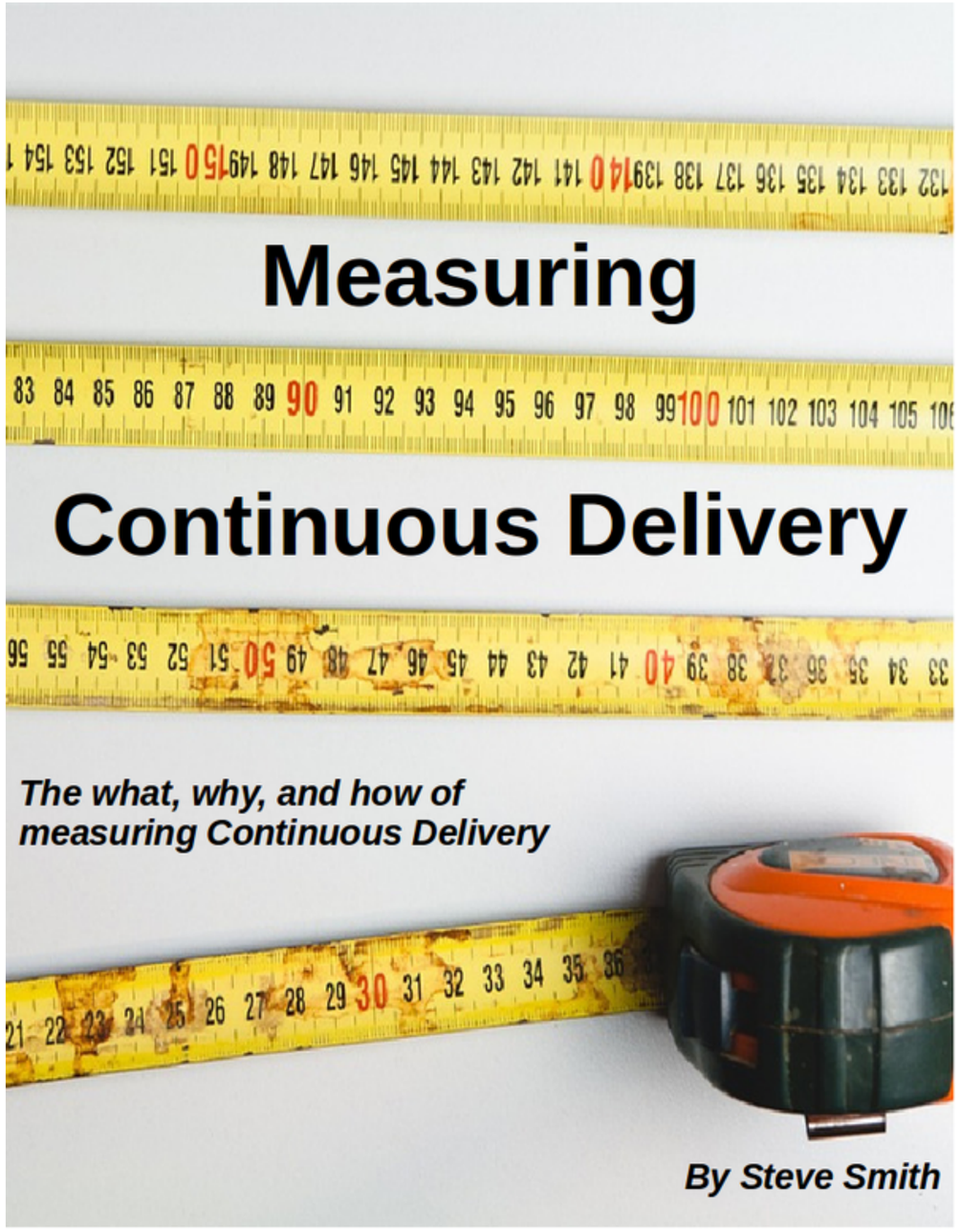
PAIR PROGRAMMING

@michieltcs


BOY SCOUT RULE

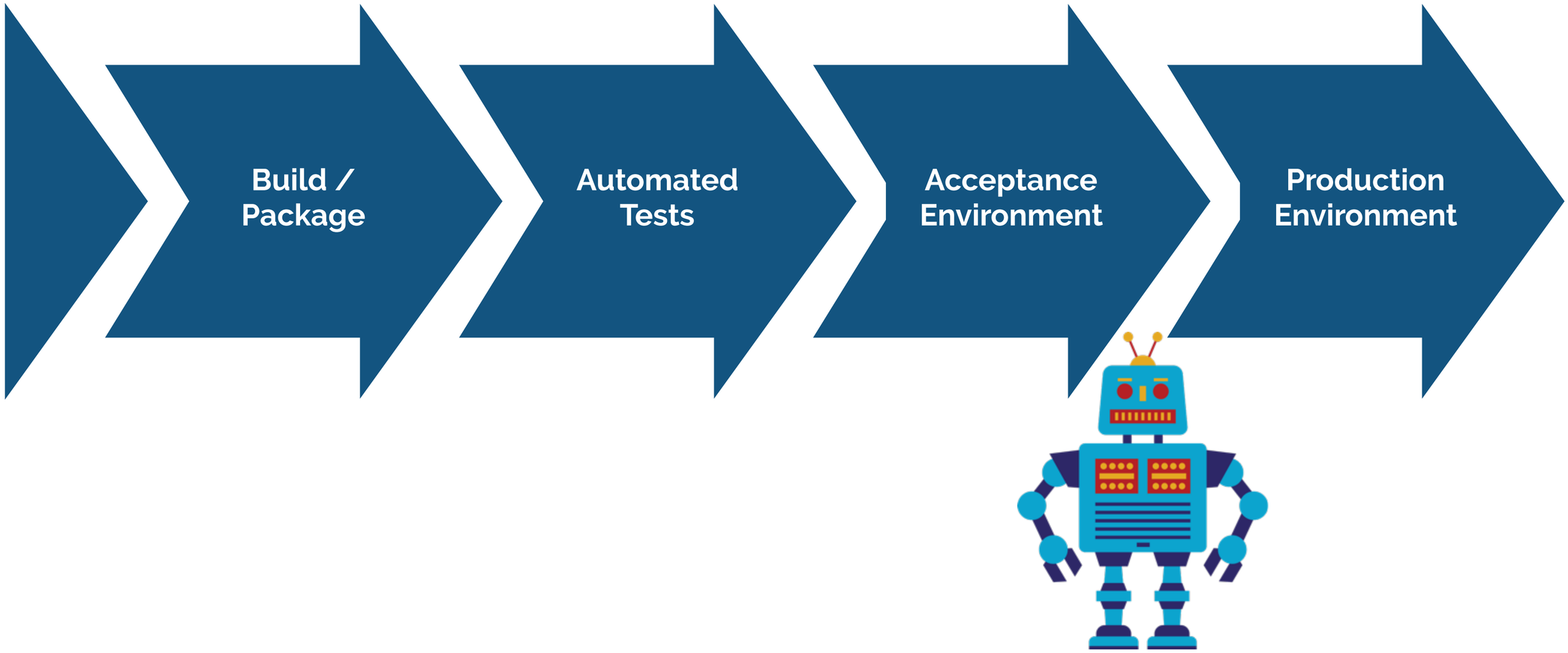
NO MORE RELEASE BRANCHES


Hallway monitor
Created by [redacted]
Last edited 4/23/18
powered by INSIGHTS
Default 30m 60m 6h 1d 7d Custom
End-user Apdex Score
Since 3 hours ago
Backend throughput
Since 3 hours ago
Page load time (Avg/95th)
Since 3 hours ago
Backend response time
Since 3 hours ago
Backend errors
Since 3 hours ago
Queue messages ready
Since 3 hours ago
Server memory usage
Since 3 hours ago
elasticsearch jobs web varnish rabbitmq mysql
Server CPU usage
Since 3 hours ago
web jobs rabbitmq mysql varnish elasticsearch
Delete
Permalink


Measuring
Continuous Delivery
The what, why, and how of measuring Continuous Delivery
By Steve Smith


CI/CD metrics for [redacted]
successful builds
failed builds
build failure rate
std dev
median recovery time
std dev
median duration
std dev
median recovery time
CI/CD metrics for [redacted]
successful builds
failed builds
build failure rate
std dev
median recovery time
std dev
median duration
std dev
median recovery time
CI/CD metrics for [redacted]


PHASE 2:
SINGLE PIPELINE

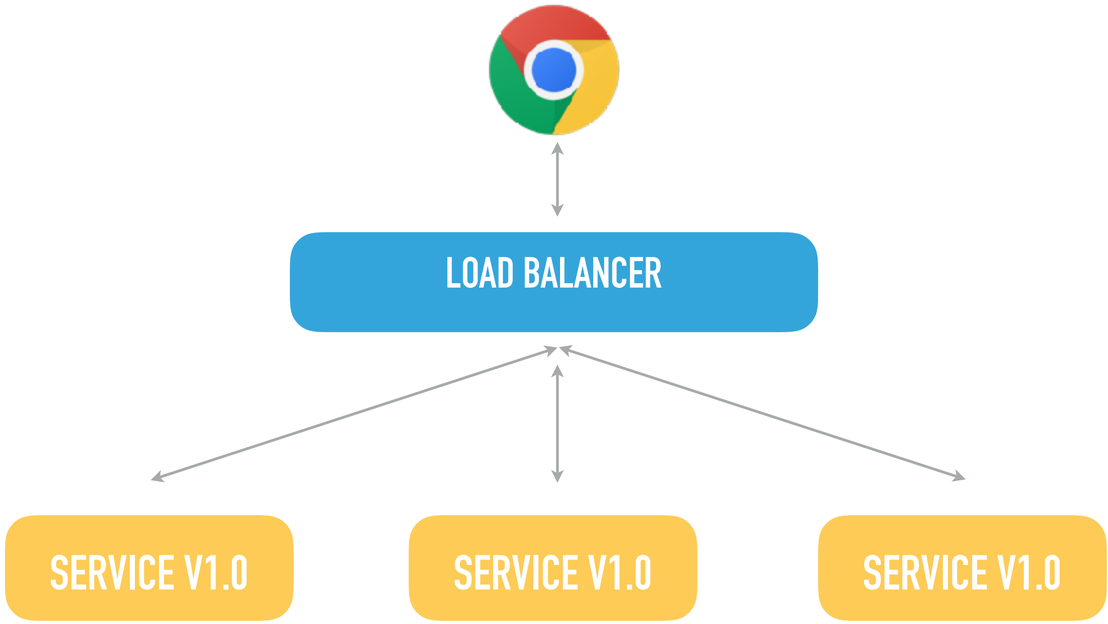

Build /
Package
Automated
Tests
Acceptance
Environment
Production
Environment

CONTINUOUS DEPLOYMENT

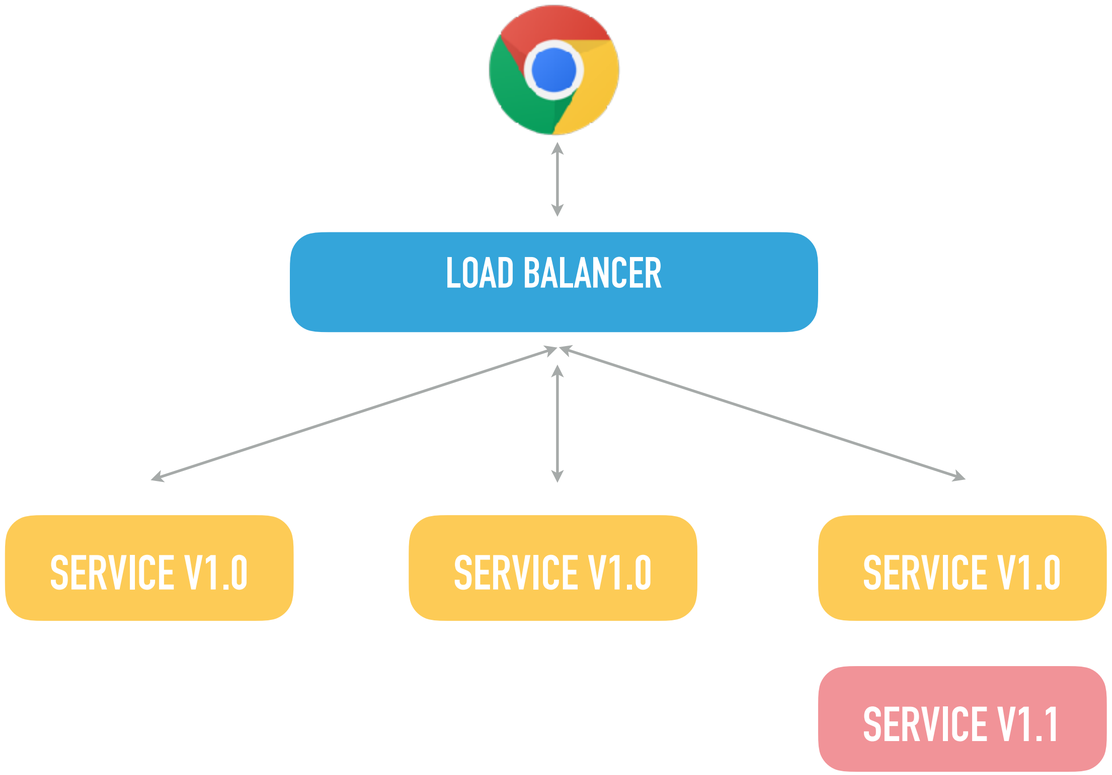
ZERO DOWNTIME DEPLOYS


ROLLING
DEPLOYS

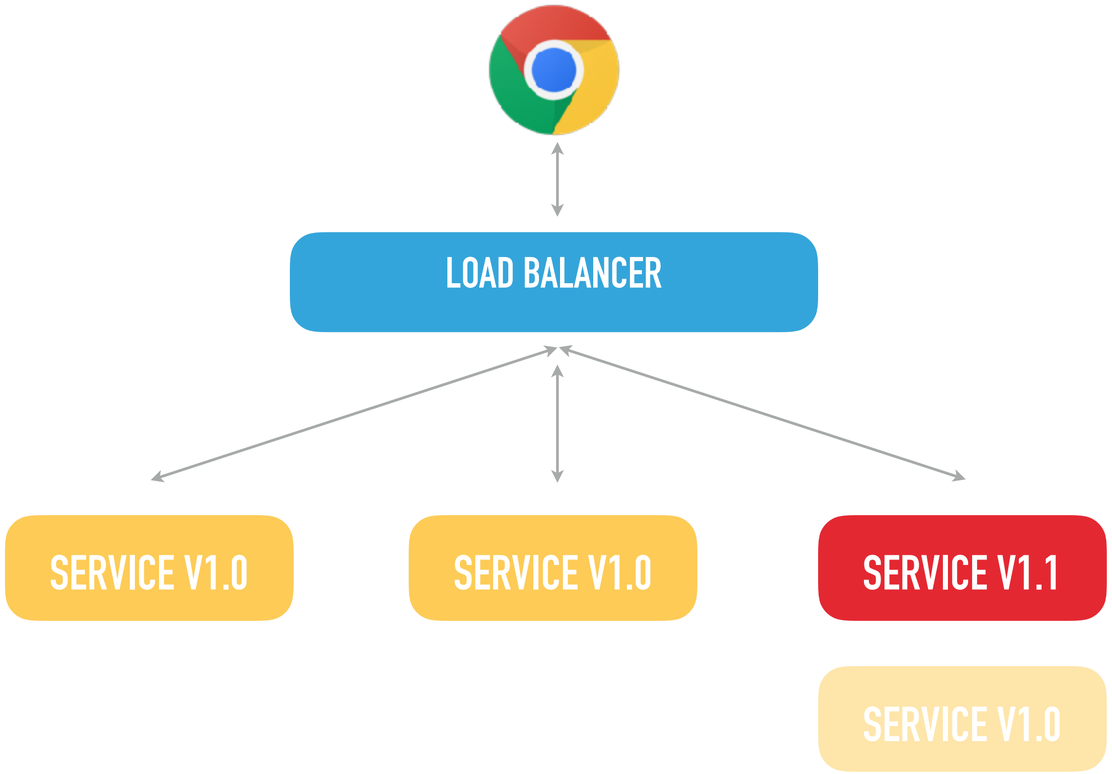
ROLLING UPDATE



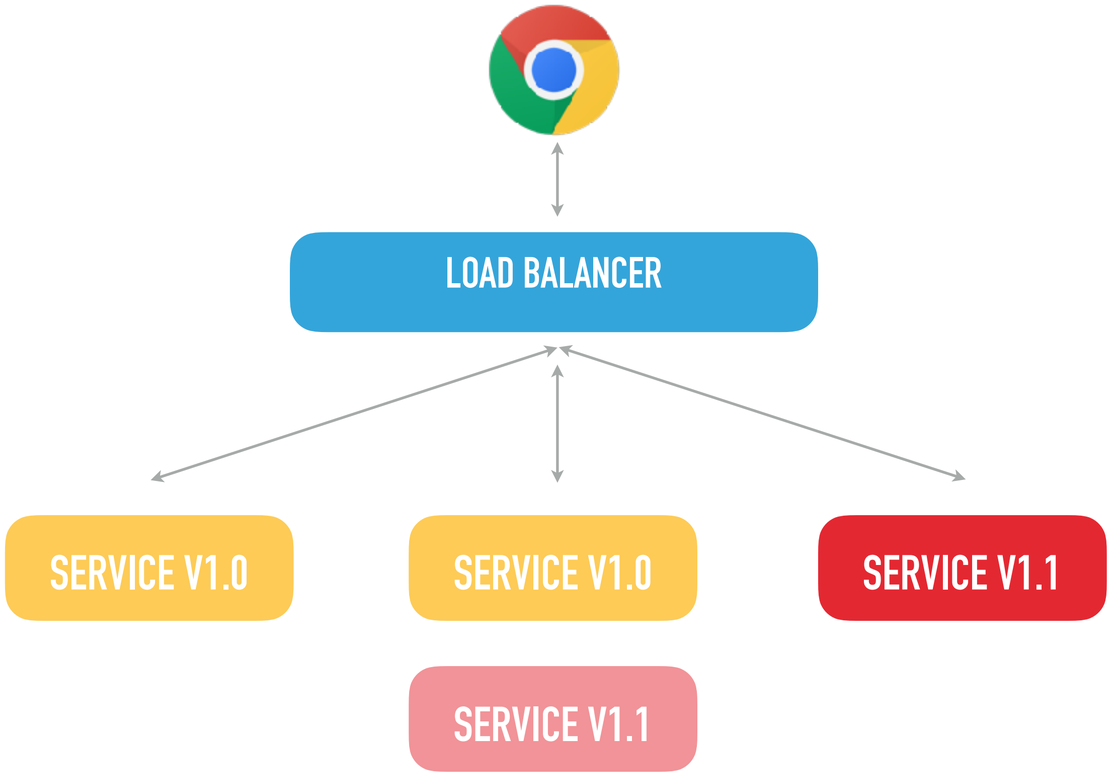
ROLLING UPDATE



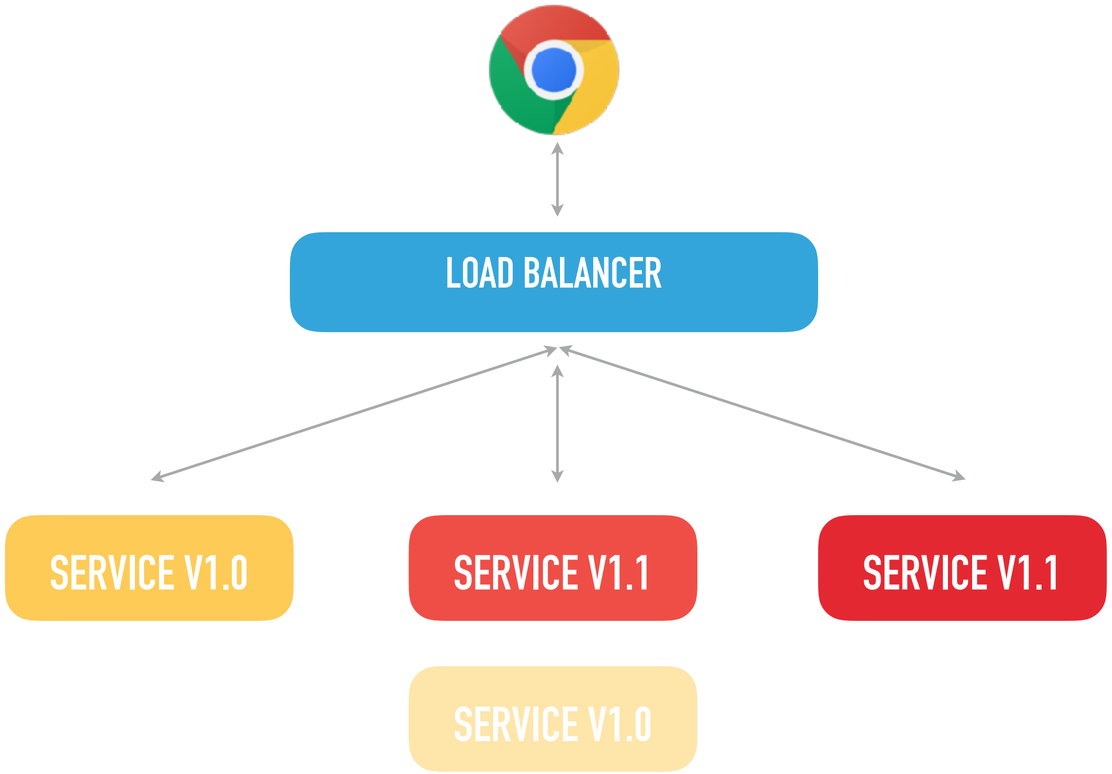
ROLLING UPDATE



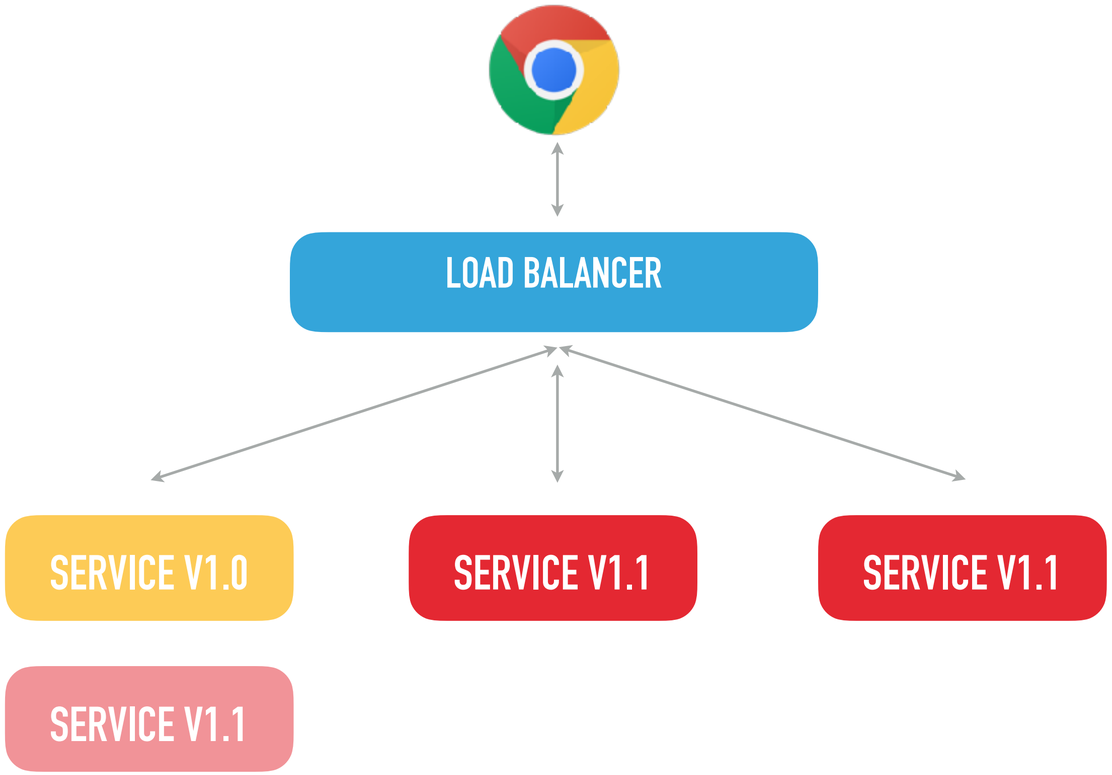
ROLLING UPDATE



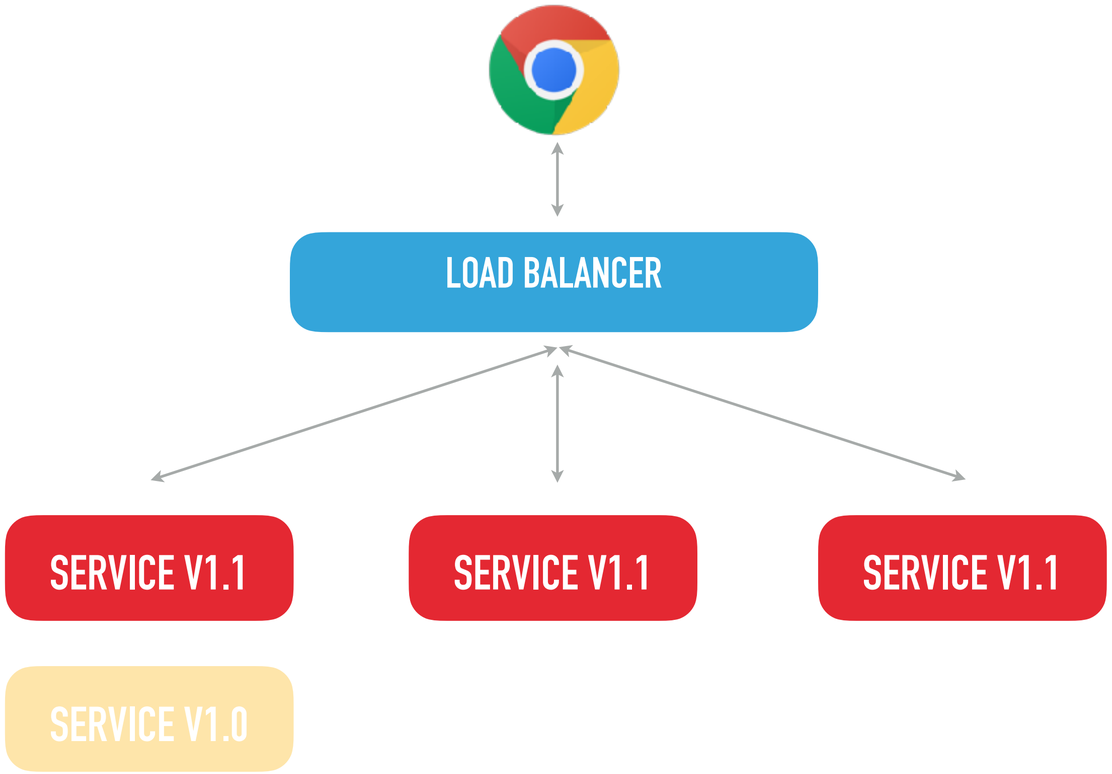
ROLLING UPDATE



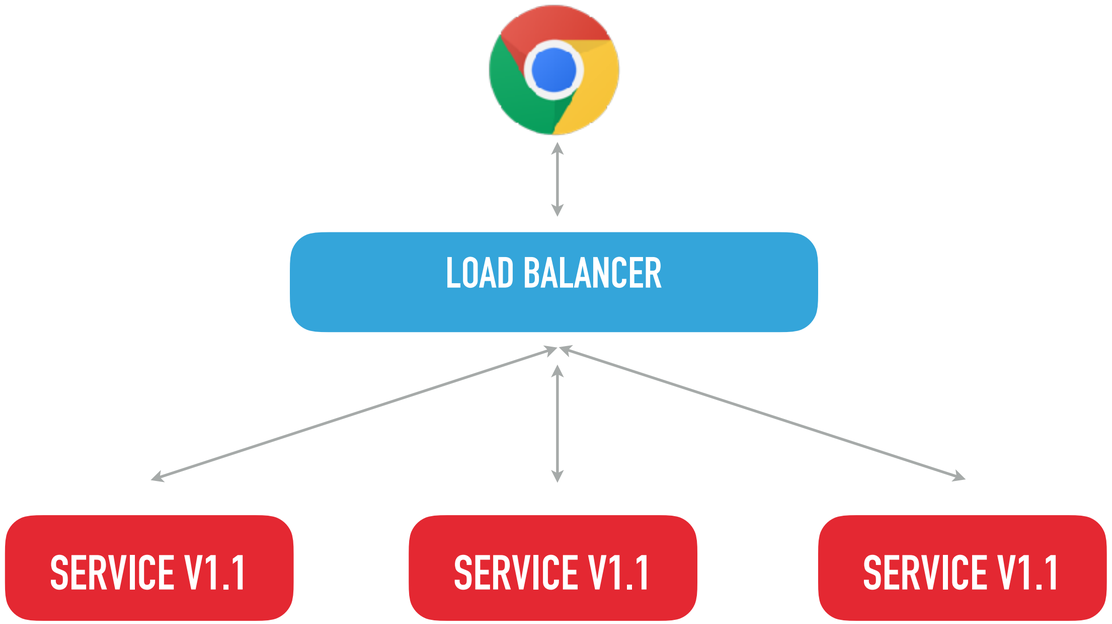
ROLLING UPDATE



ROLLING UPDATE



ROLLING UPDATE



PIPELINE FAILURES

FLAKY TESTS

TIMEOUTS

NETWORK STABILITY ISSUES

EXTERNAL DEPS IN TESTS

BRITTLE INFRASTRUCTURE

IF YOU CAN'T TRUST IT - REMOVE IT


STOP THE LINE

EXTREME FEEDBACK

@michieltcs

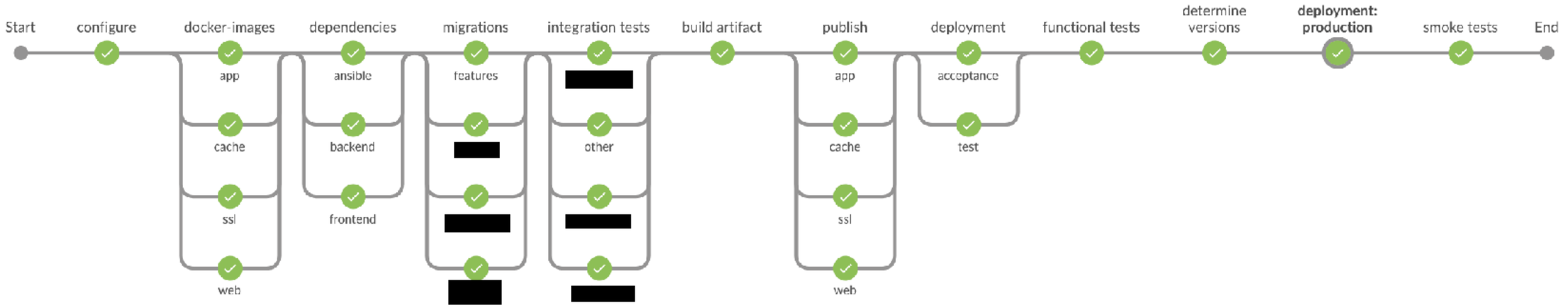
[redacted] 1087

Pipeline | Changes | Tests | Artifacts | Login

Branch: –
Commit: –

21m 13s
42 minutes ago

Changes by [redacted]
Started by an SCM change



Steps deployment: production

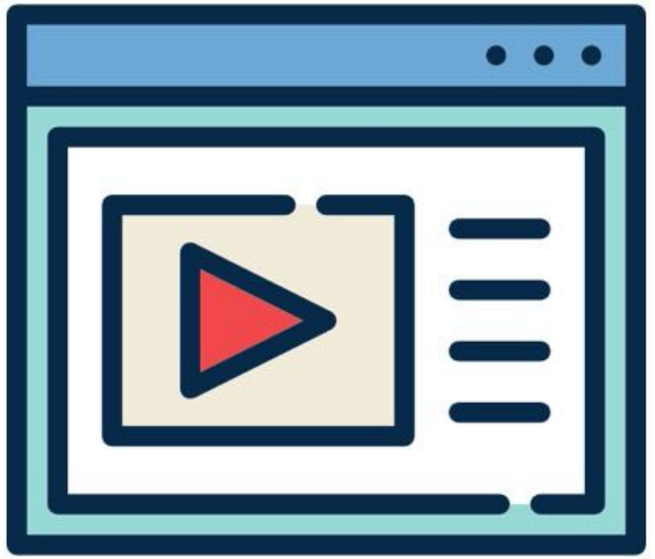
✓	› ansible-playbook --verbose ansible/galaxy_roles.yml — Shell Script	6s
✓	› Shell Script	2m 28s
✓	› Shell Script	3m 19s

RELEASE TIMING?

FEATURE TOGGLES

DECOUPLE DEPLOYMENTS FROM RELEASES

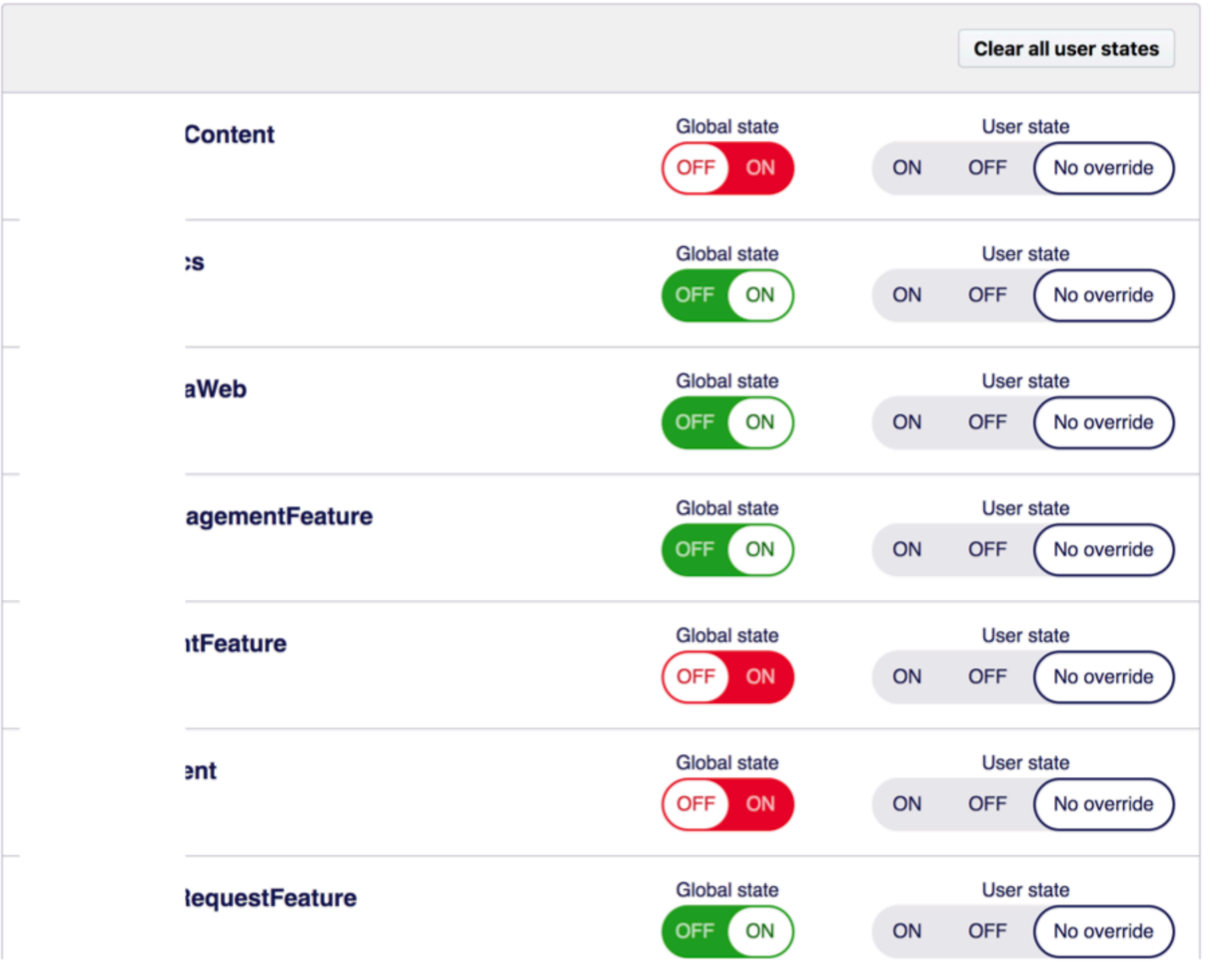
New Feature



Feature Flag or Toggle

Consumers

Feature flag dashboard



CAUTION: FEATURE TOGGLE DEBT

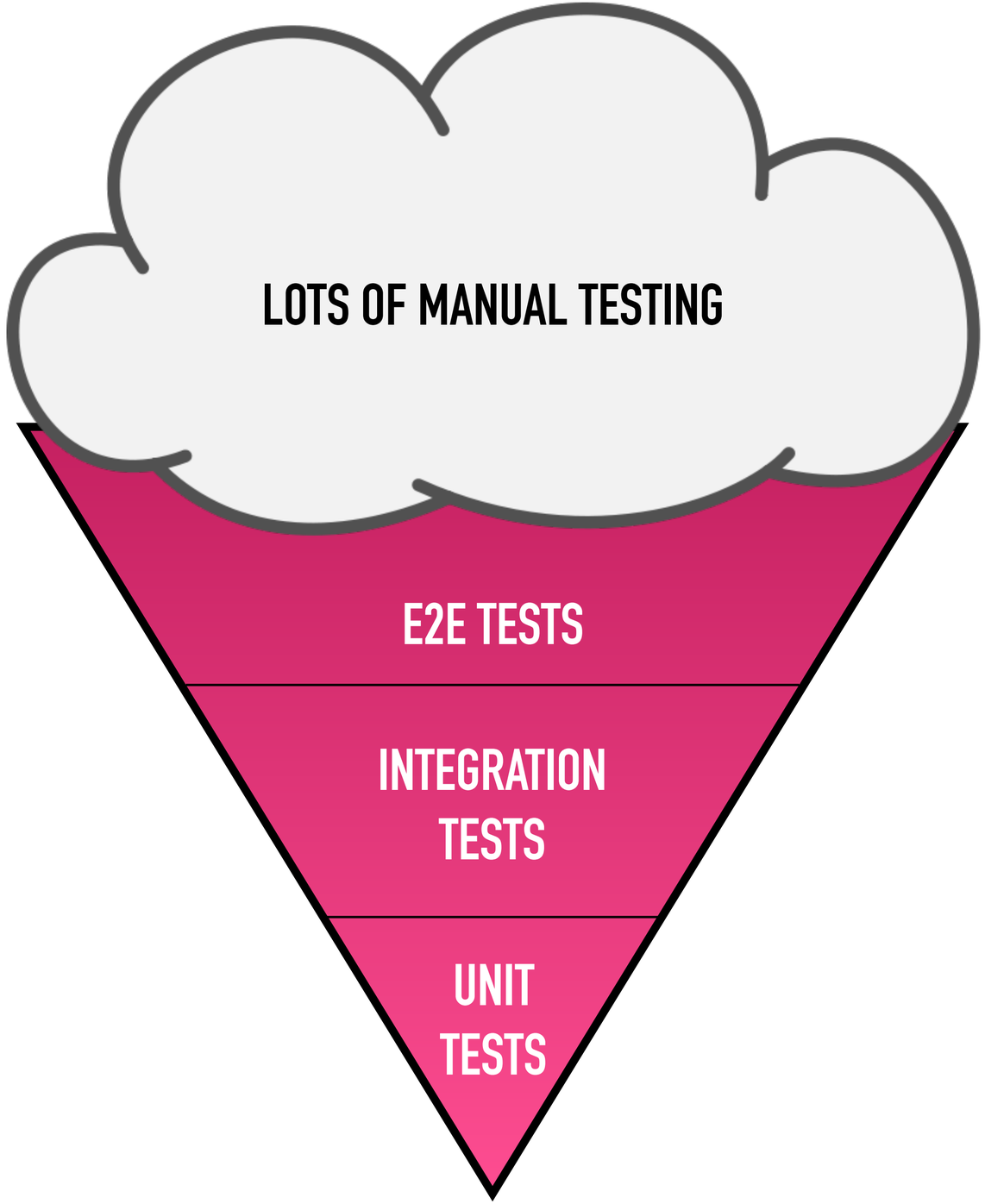
PHASE 3: IMPROVEMENTS & TESTING

PIPELINE SPEED

BUY A BIGGER BOX

CLEVER PIPELINE IMPROVEMENTS: PARALLELIZE

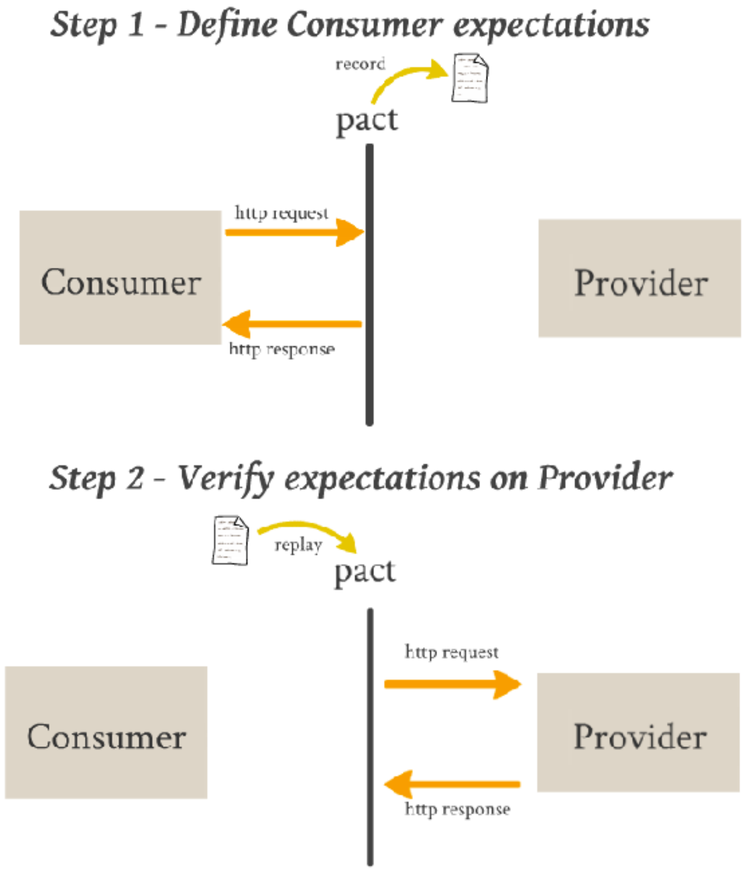
TESTING IMPROVEMENTS


LOTS OF MANUAL TESTING
E2E TESTS
INTEGRATION TESTS
UNIT TESTS

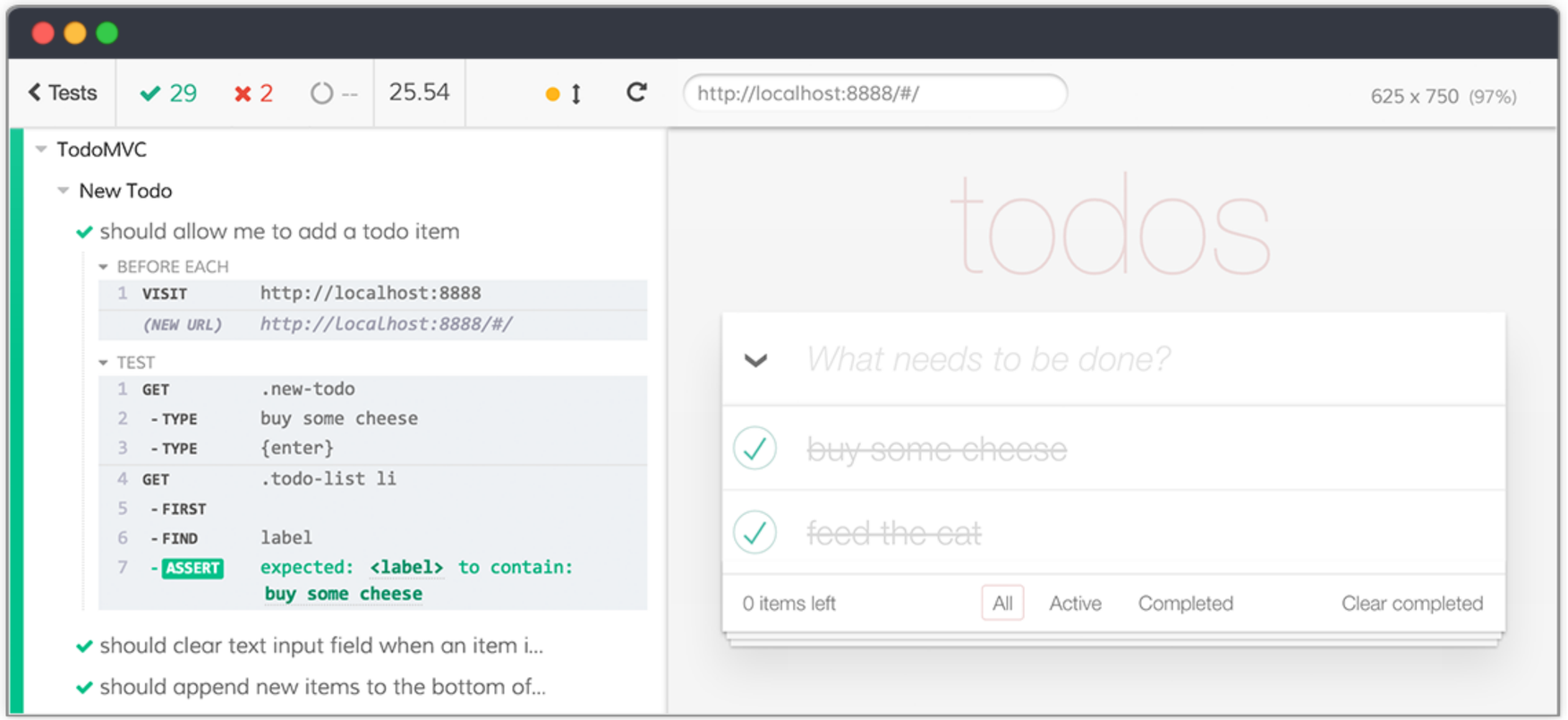
QUALITY ENGINEERING

@michieltcs

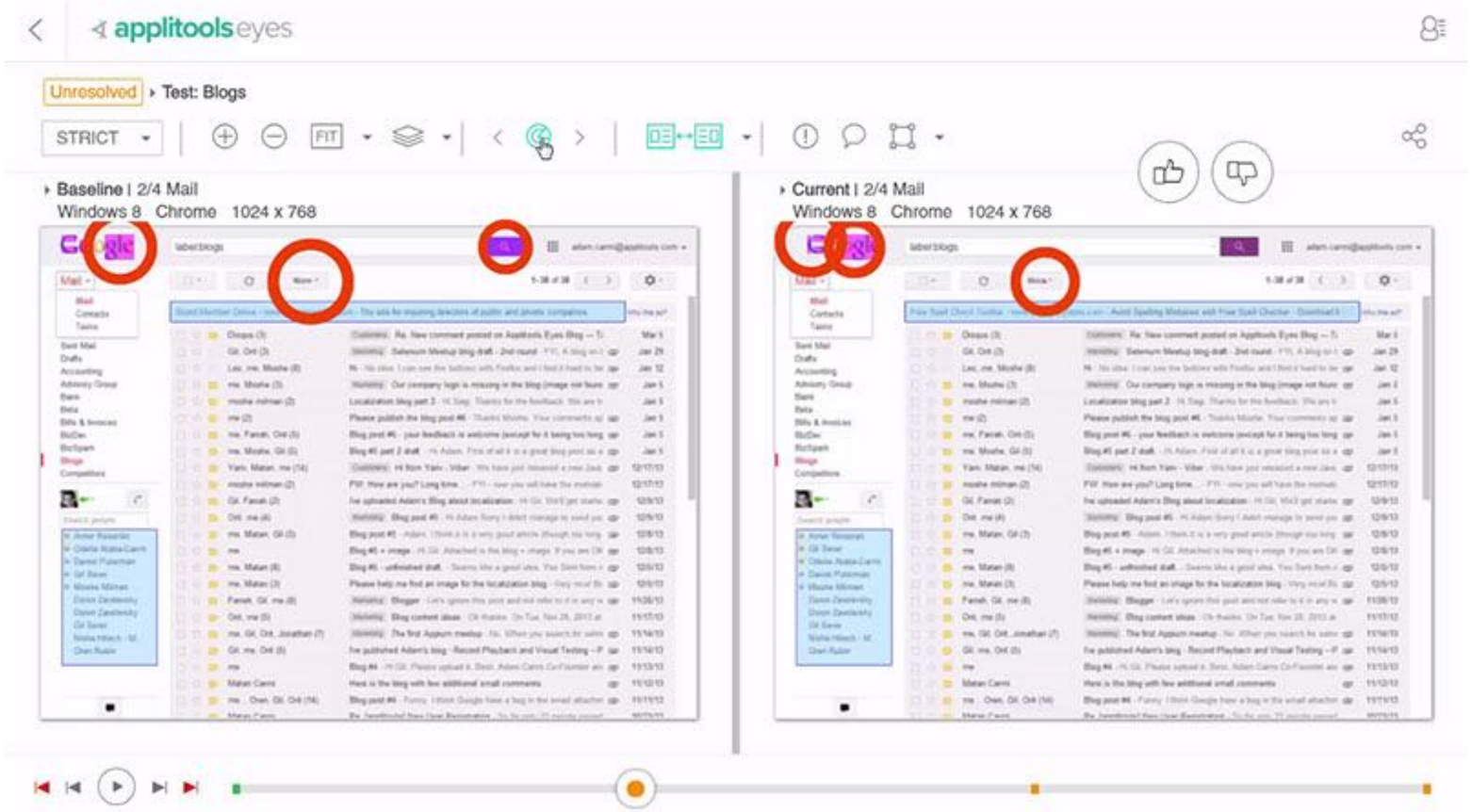
AUTOMATION



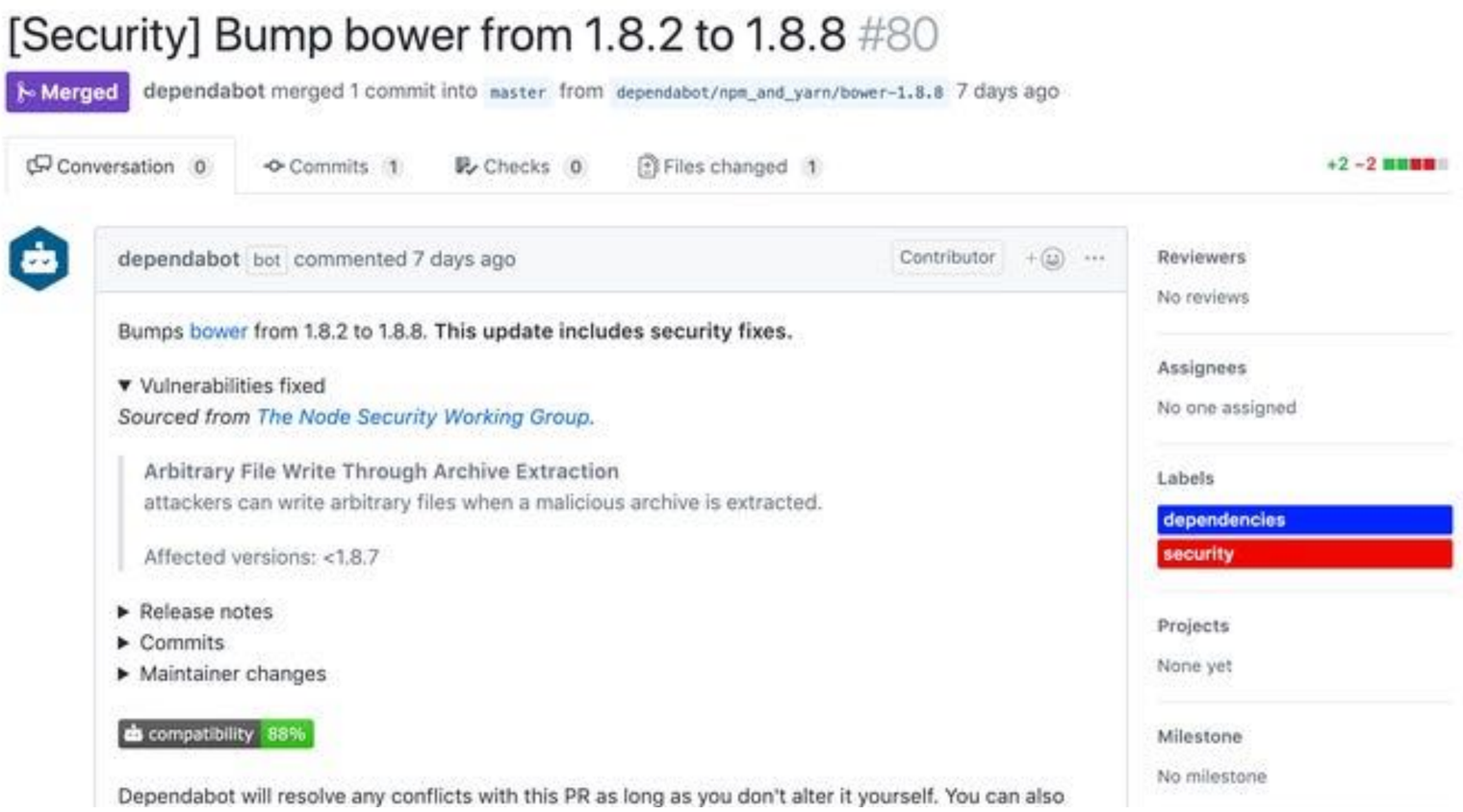
CONTRACT TESTING


Tests
29
2
--
25.54
http://localhost:8888/#/
625 x 750 (97%)
TodoMVC
New Todo
should allow me to add a todo item
BEFORE EACH
1 VISIT http://localhost:8888
(NEW URL) http://localhost:8888/#/
TEST
1 GET .new-todo
2 - TYPE buy some cheese
3 - TYPE {enter}
4 GET .todo-list li
5 - FIRST
6 - FIND label
7 - ASSERT expected: <label> to contain: buy some cheese
should clear text input field when an item i...
should append new items to the bottom of...
todos
What needs to be done?
buy some cheese
feed the cat
0 items left
All
Active
Completed
Clear completed

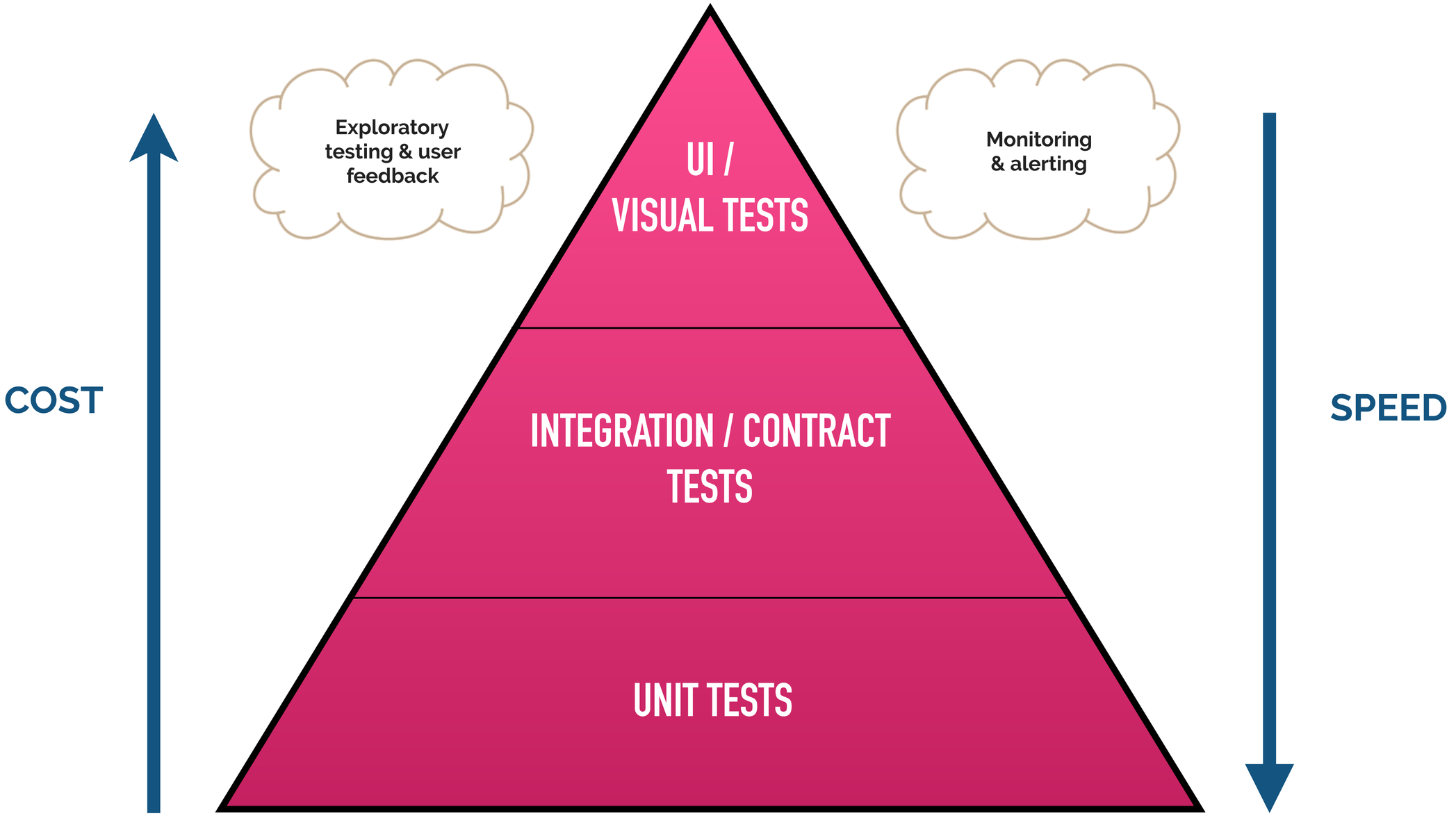
UI TESTING



VISUAL TESTING



DEPENDENCY SCANNING


Exploratory testing & user feedback
Monitoring & alerting
UI / VISUAL TESTS
INTEGRATION / CONTRACT TESTS
UNIT TESTS
COST
SPEED

RESULTS & TAKEAWAYS

LEARNINGS

PIPELINE SPEED & STABILITY

TEST THE PIPELINE

WAY OF WORKING & MINDSET

VISUAL TESTING TOOLS

SOME STATS

~4000 TESTS PER DEPLOYMENT

~10 MINUTES PER PIPELINE RUN

~1.5 HOURS FROM DEV TO PROD

IN 6 MONTHS: MORE THAN 5000 DEPLOYMENTS

THANK YOU FOR LISTENING!

@michieltcs / michiel@michielrook.nl

www.michielrook.nl