

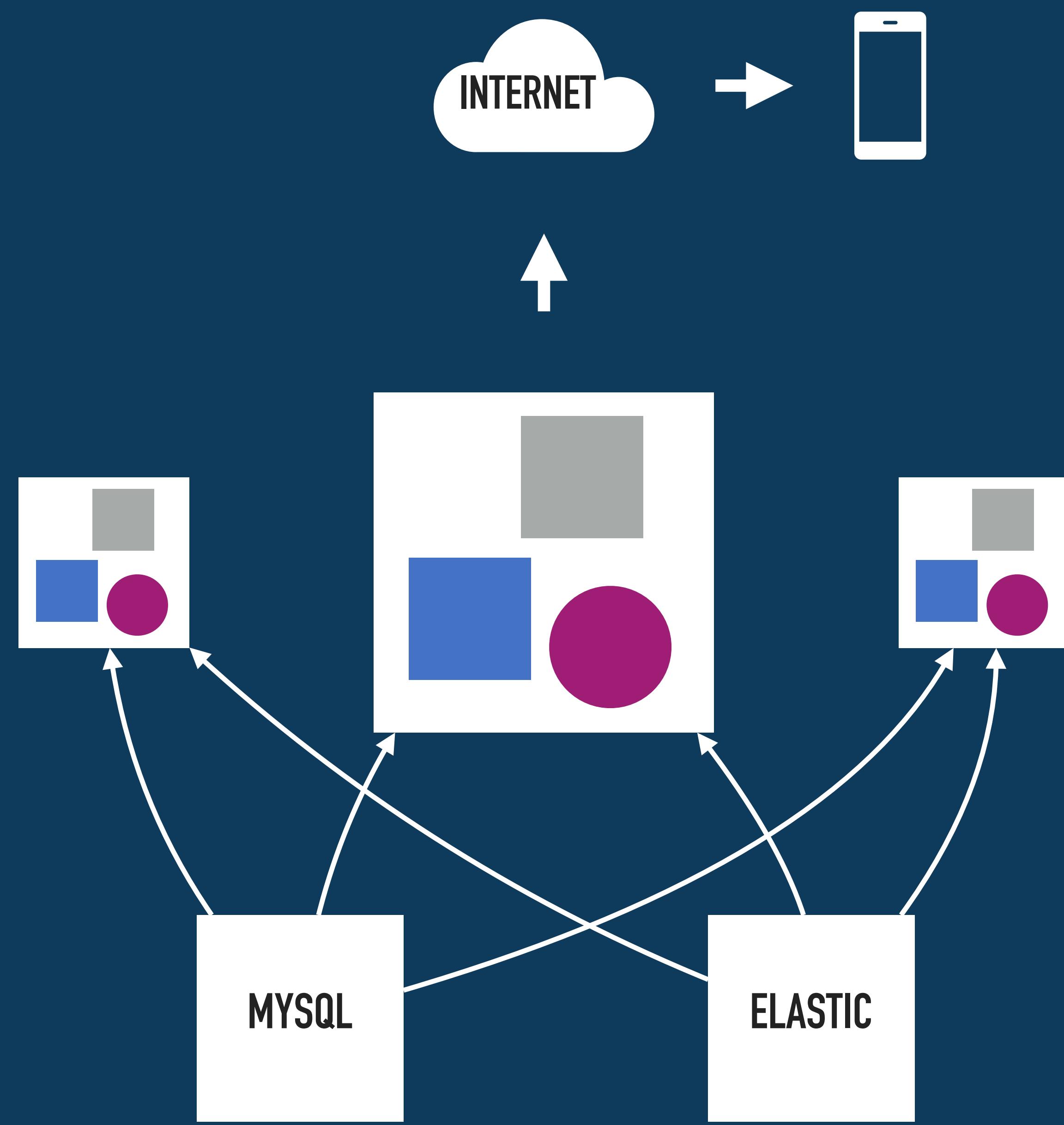


REAL WORLD CD LEARN, ADAPT & IMPROVE

MICHAEL ROOK
@MICHAELTCS

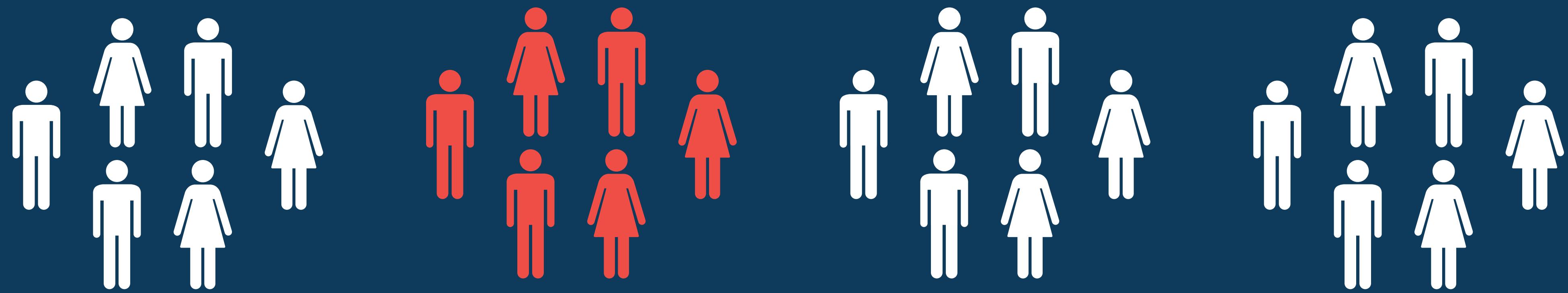
A photograph of several modern skyscrapers in a city skyline under a blue sky with scattered clouds. The buildings are primarily glass and steel, reflecting the light. One building has a curved facade. In the foreground, there is a large, semi-transparent white text overlay.

LARGE SEMI-GOV
INNL





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 branch 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 branch 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.

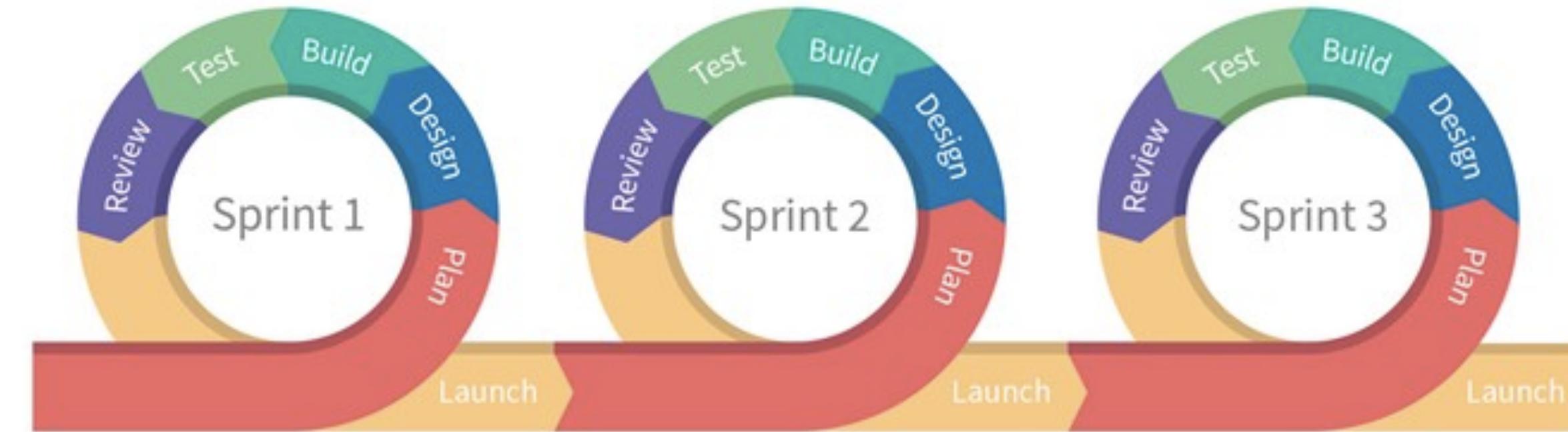
```
git checkout master  
git merge hotfix/fix-some-crucial-bug
```

- 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



every
2 weeks



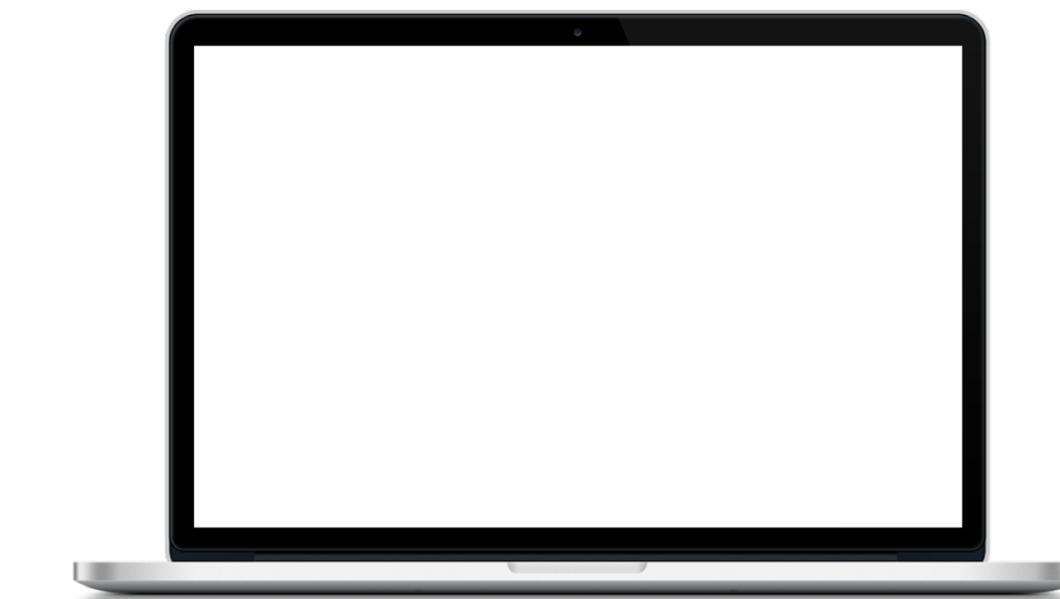
Thursday
(sprint end)



Monday
(sprint start)



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

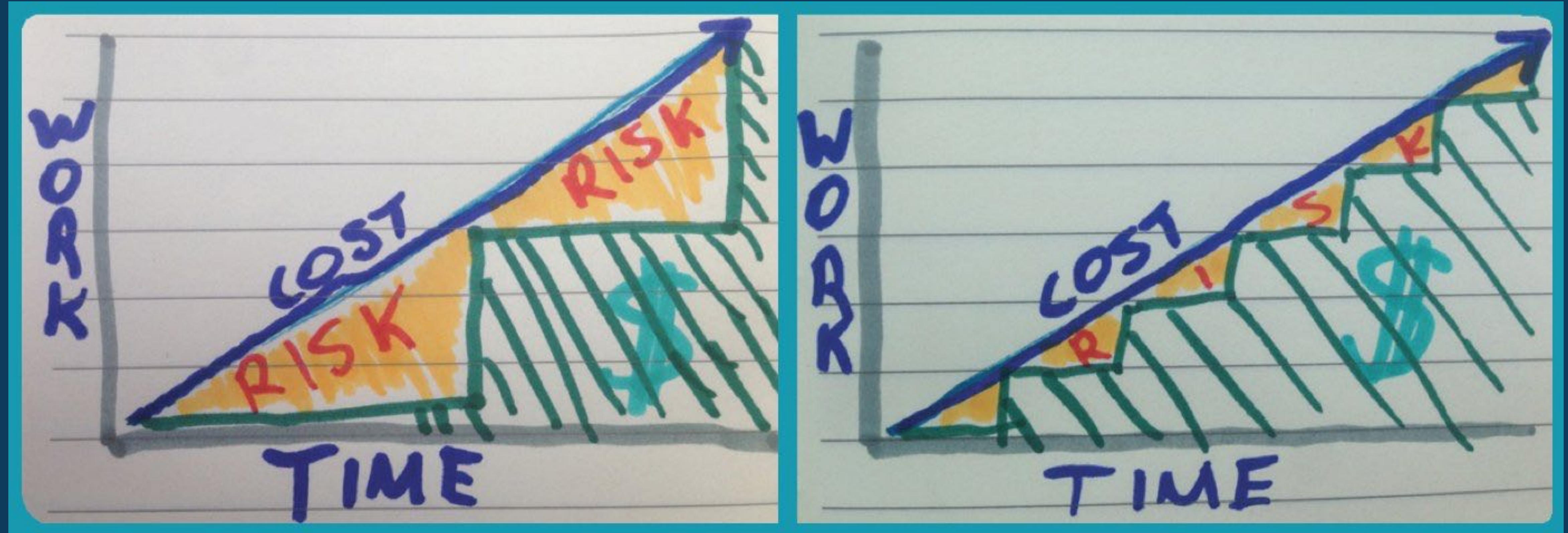
BIG STEPS

FAIL BIG

SMALL STEPS

FAIL SMALL

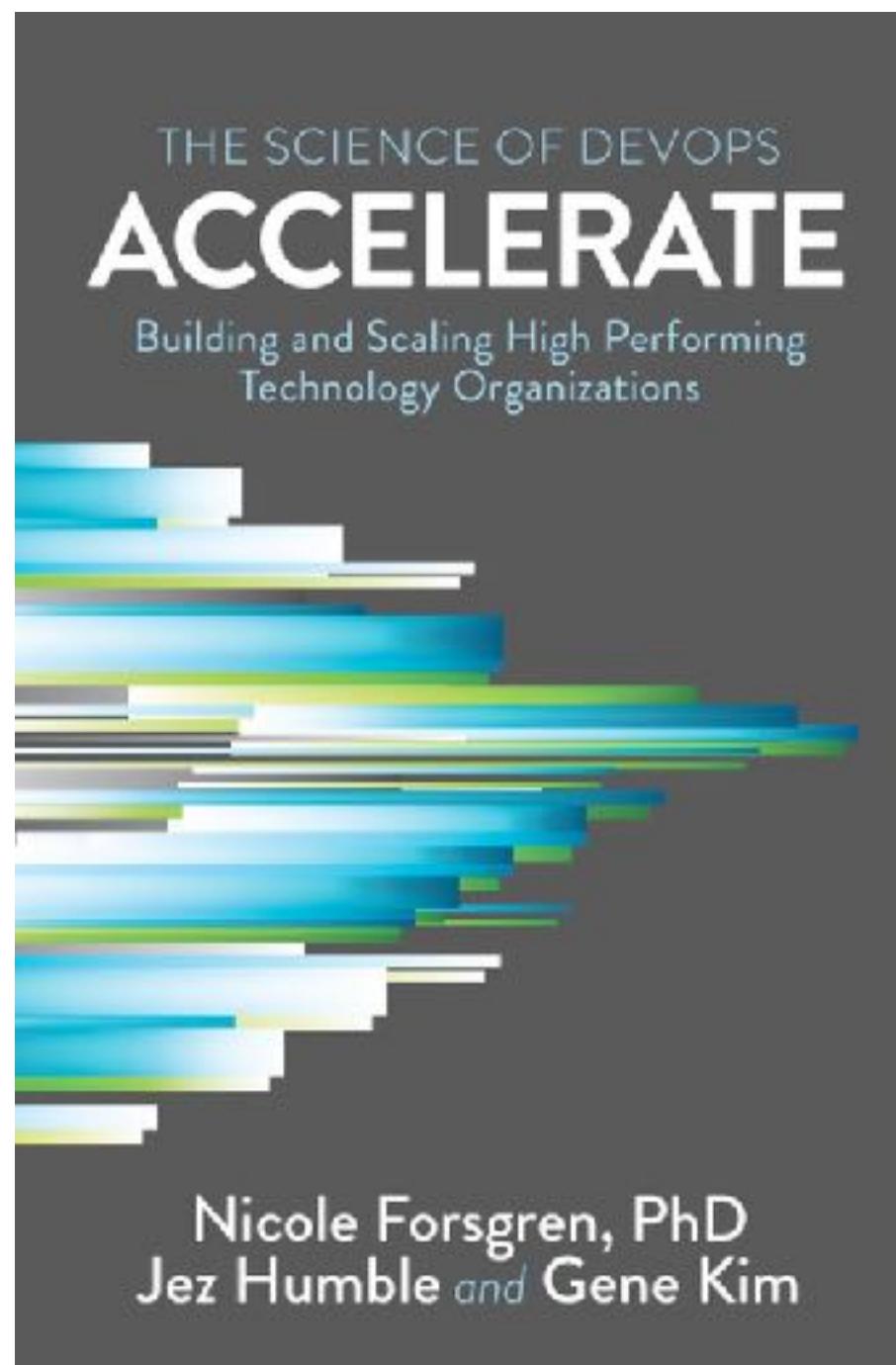
IF IT HURTS
DO IT MORE OFTEN



\$ = 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

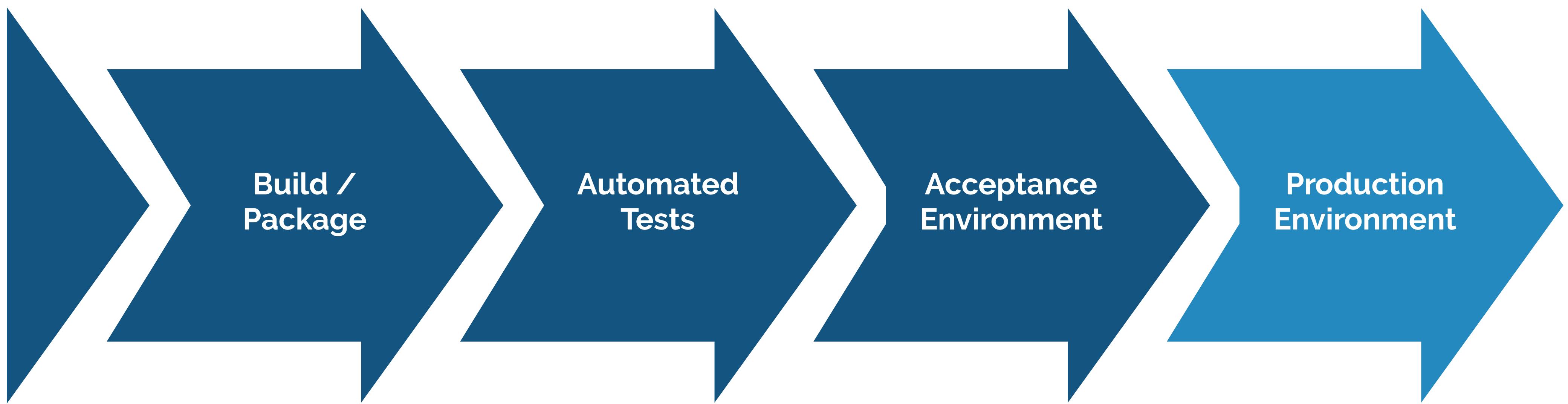
IMPROVEMENTS & TESTING

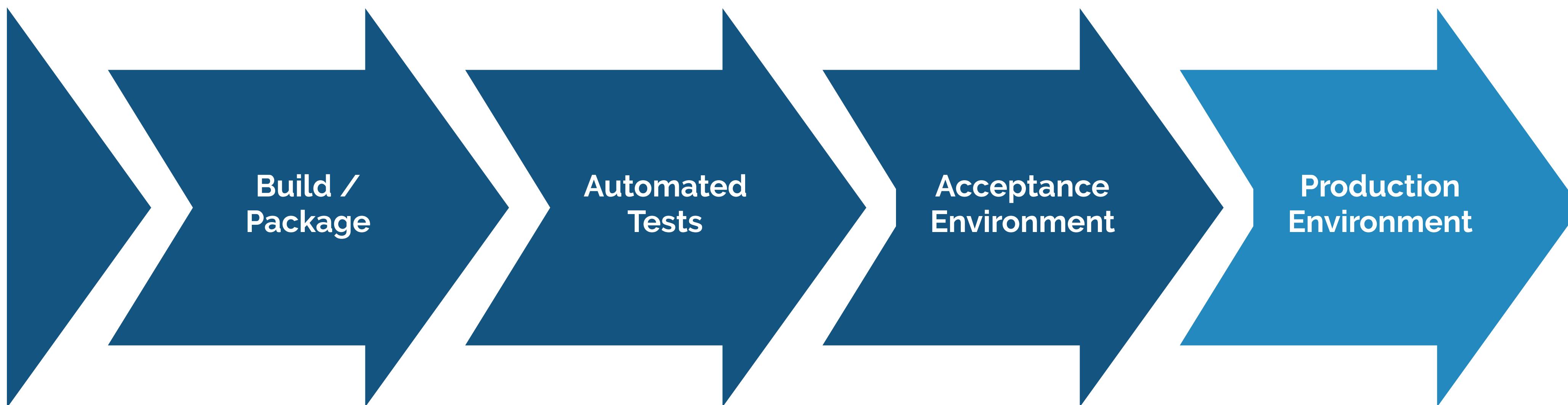
A dynamic street performance by a band of drummers. Several musicians are shown in mid-motion, playing large red and black bass drums. The drum shells feature the brand name "bloko" and the website "blokodelvalle.com". The performers are dressed in dark clothing with red and white accents, some wearing face paint and headbands. The background shows a yellow building with a dark wooden door and a metal shutter. The overall atmosphere is energetic and rhythmic.

PHASE 1: INCREASING RELEASE CADENCE

WEEKLY RELEASE

AUTOMATION





PIPELINE AS CODE

The word "PIPELINE AS CODE" is rendered in large, bold letters. Each letter contains a different snippet of CI/CD configuration code, such as Jenkins Pipeline, Ansible Playbook, or Dockerfile snippets, illustrating the concept of treating pipelines as code.

- P: pipeline {
- I: stages {
- E: stage('Run tests') {
- L: sh "gradle check"
- N: }
- A: }
- S: stage('Build docker image') {
- H: build -t jobservice:\$env.BUILD_NUMBER . "
- D: phpunit --coverage-clover ./clover.xml --coverage-text ./coverage.txt
- E: stage('Deploy staging') {
- N: sh "ansible-playbook -e BUILD=\$env.BUILD_NUMBER -i staging deploy.yml"
- P: }
- E: stage('Deploy production') {
- N: sh "ansible-playbook -e BUILD=\$env.BUILD_NUMBER -i prod deploy.yml"
- D: deployment: env.BUILD_NUMBER
- E: stage: deploy
- S: script:
- M: - make deploy
- O: only:
- M: - master
- A: }
- N: }
- S: }

jenkins #696 Michiel

Secure | https://jenkins. [REDACTED] ⋮

✓ [REDACTED] 696 Pipeline Changes Tests Artifacts Login X

Branch: — 20m 3s Changes by [REDACTED]
Commit: — 12 minutes ago Started by an SCM change

Start configure dependencies phpunit tests artifact(ory) deployment cucumber tests trigger End

backend
frontend

acceptance
test

Steps trigger

✓ > Building [REDACTED] deploy to production <1s
✓ > nightwatch/report/cucumber_junit.xml – Archive JUnit-formatted test results <1s

jenkins / [REDACTED] deploy to x Michiel

Secure Login X

✓ [REDACTED] - deploy to production 66 Pipeline Changes Tests Artifacts

Branch: — 47m 22s Changes by [REDACTED]
Commit: — 2 hours ago Started by user [REDACTED]

Start decision provision: production deployment: production provision: preview deployment: preview End

Steps deployment: preview

	General SCM	deployment to preview	ansible-playbook --inventory ansible [REDACTED] . – Shell Script	2m 41s
✓ >	2s	<1s		

WAY OF WORKING



PAIR PROGRAMMING



BOY SCOUT RULE

NO MORE RELEASE BRANCHES



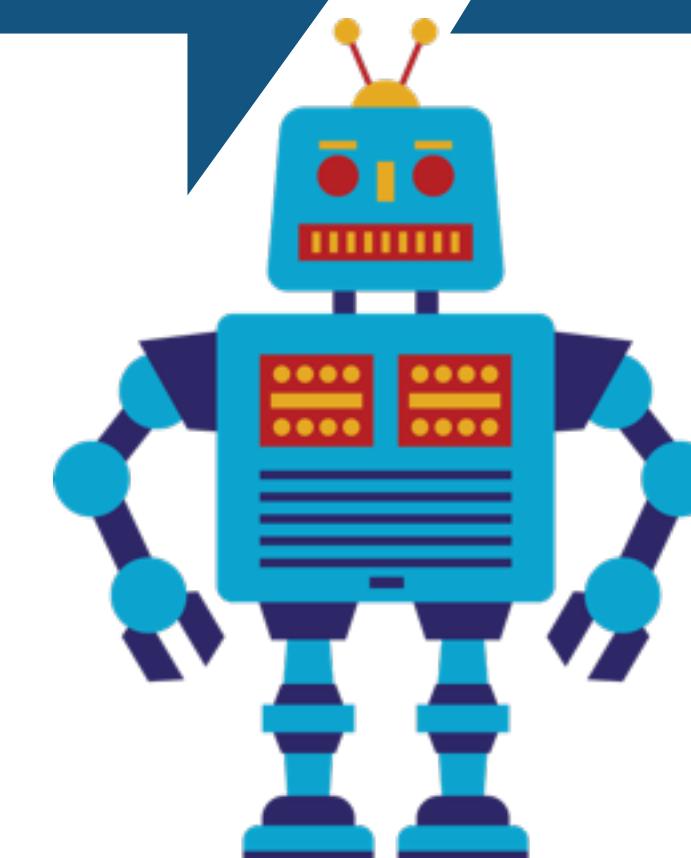
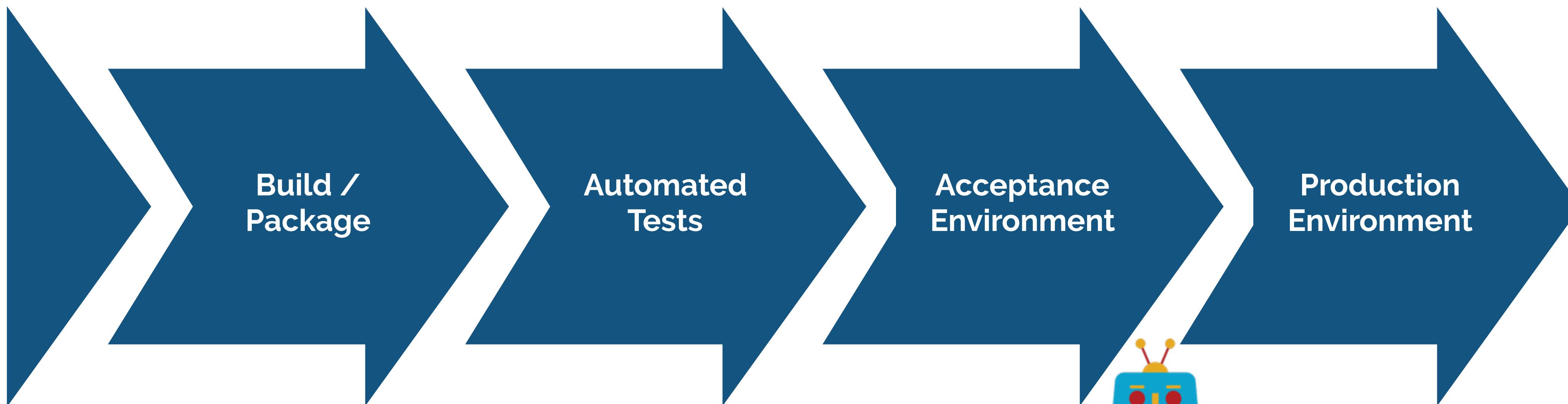
@michieltcs





A large, dark blue-grey pipeline runs diagonally across a dry, hilly landscape. The terrain is covered in sparse, brownish-green vegetation. In the background, several hills and mountains are visible under a clear blue sky with a few wispy clouds.

PHASE 2: SINGLE PIPELINE



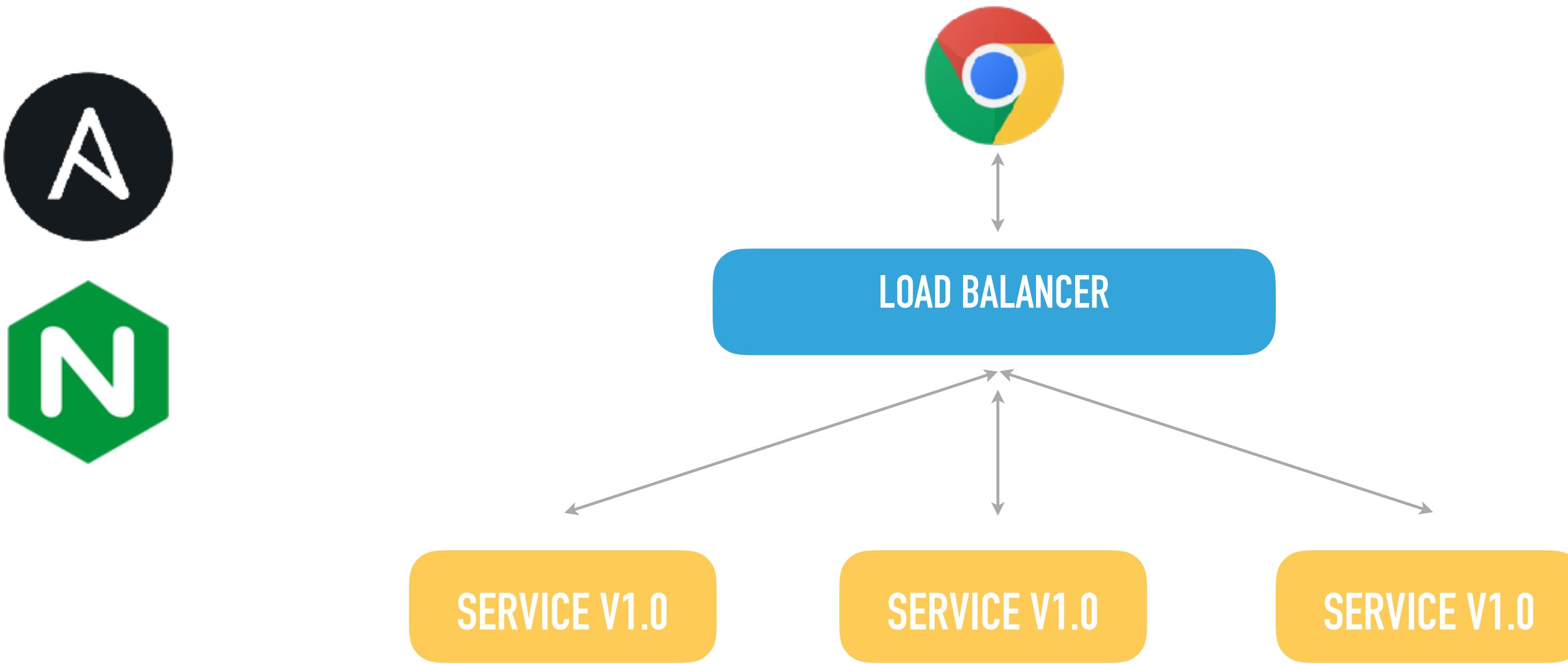
CONTINUOUS DEPLOYMENT

ZERO DOWNTIME DEPLOYS

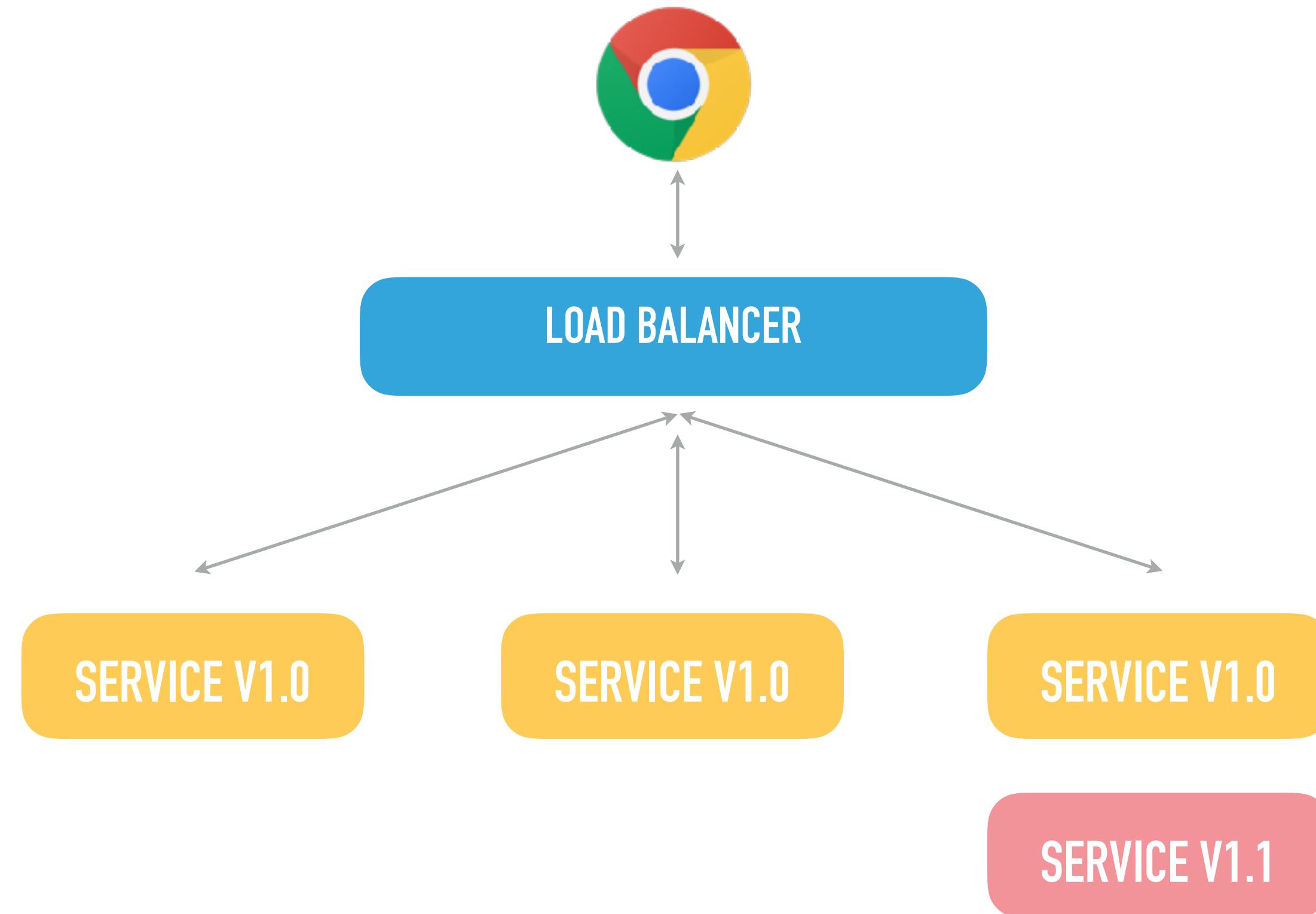
A wide-angle photograph of a traditional cheese rolling event. A long line of people, mostly men, are sliding down a steep, grassy hill. They are wearing casual clothing like t-shirts, jeans, and hoodies. Some are wearing athletic gear like shorts and athletic shoes. The hillside is covered in green grass and some small bushes. In the background, there's a dense forest of green trees. The scene is dynamic, with people at various points of their descent, some looking back and others looking forward.

ROLLING
DEPLOYS

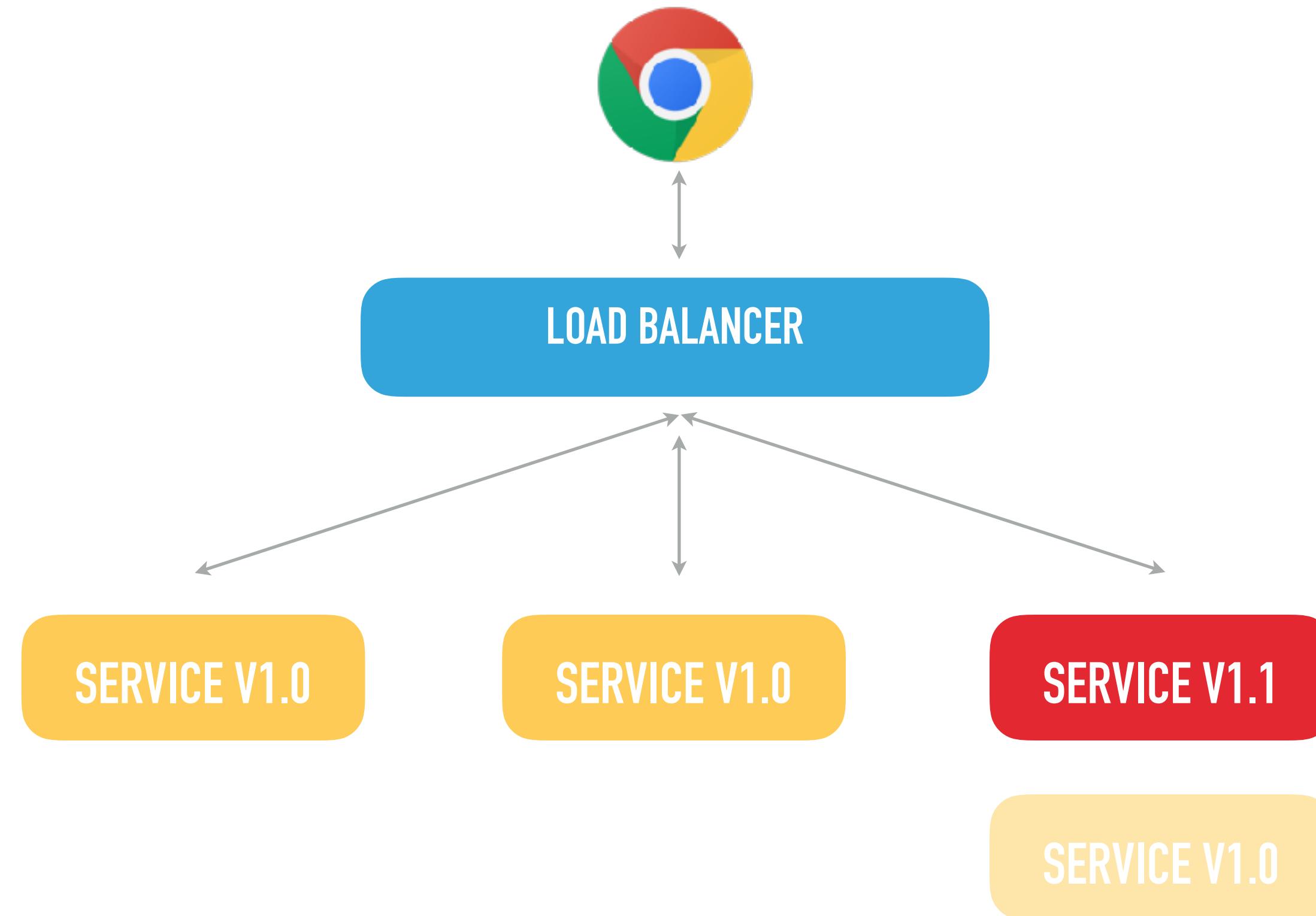
ROLLING UPDATE



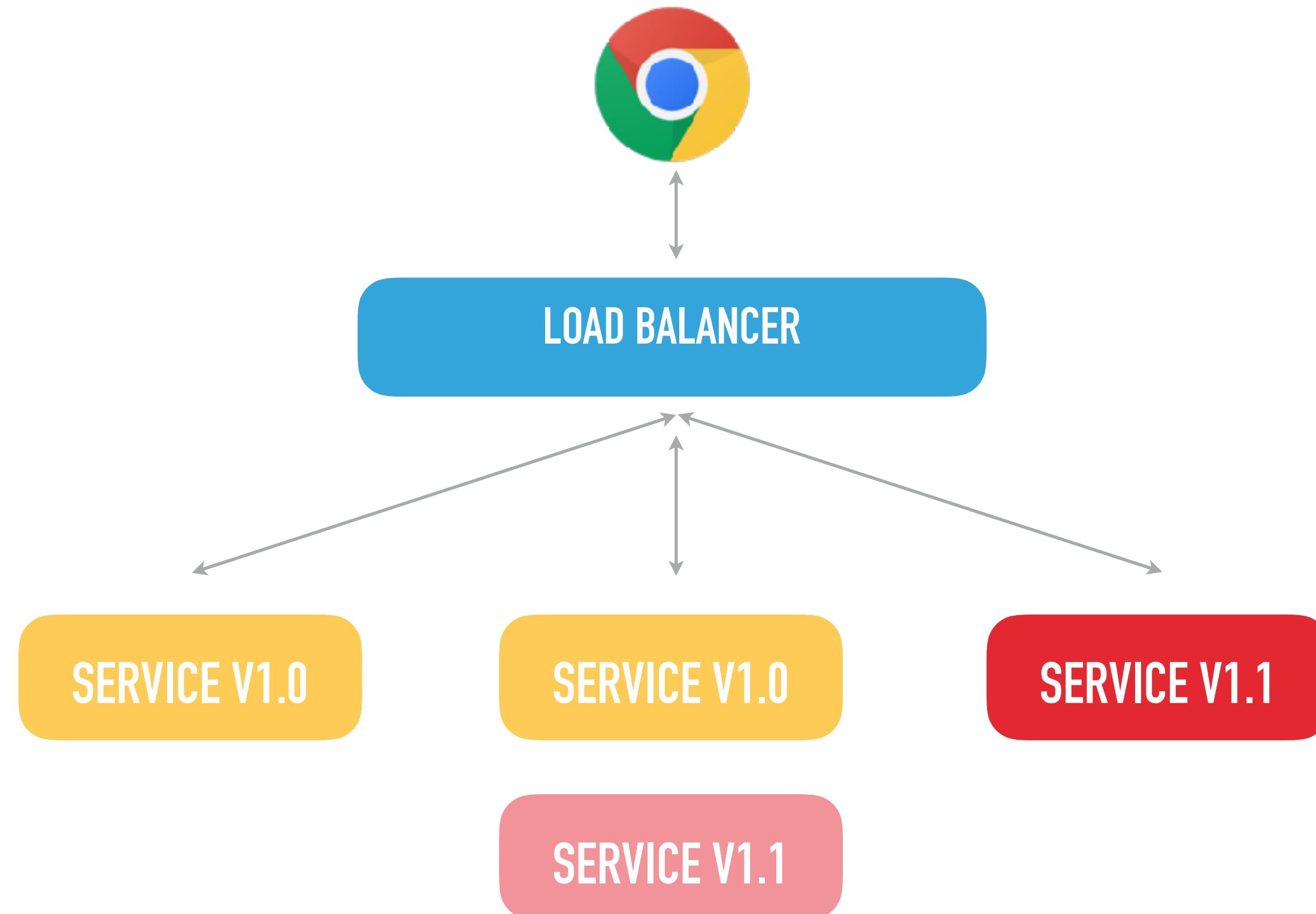
ROLLING UPDATE



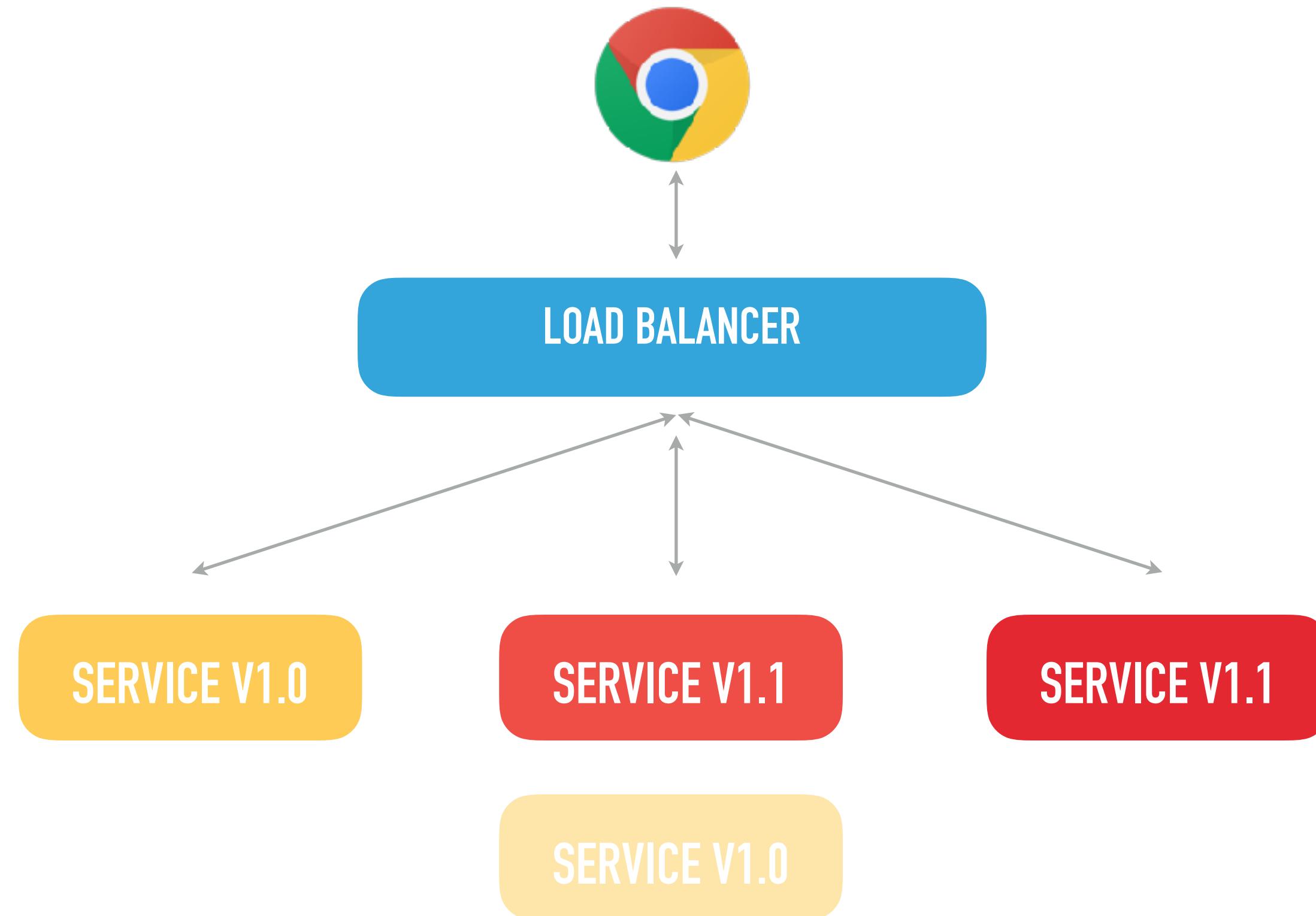
ROLLING UPDATE



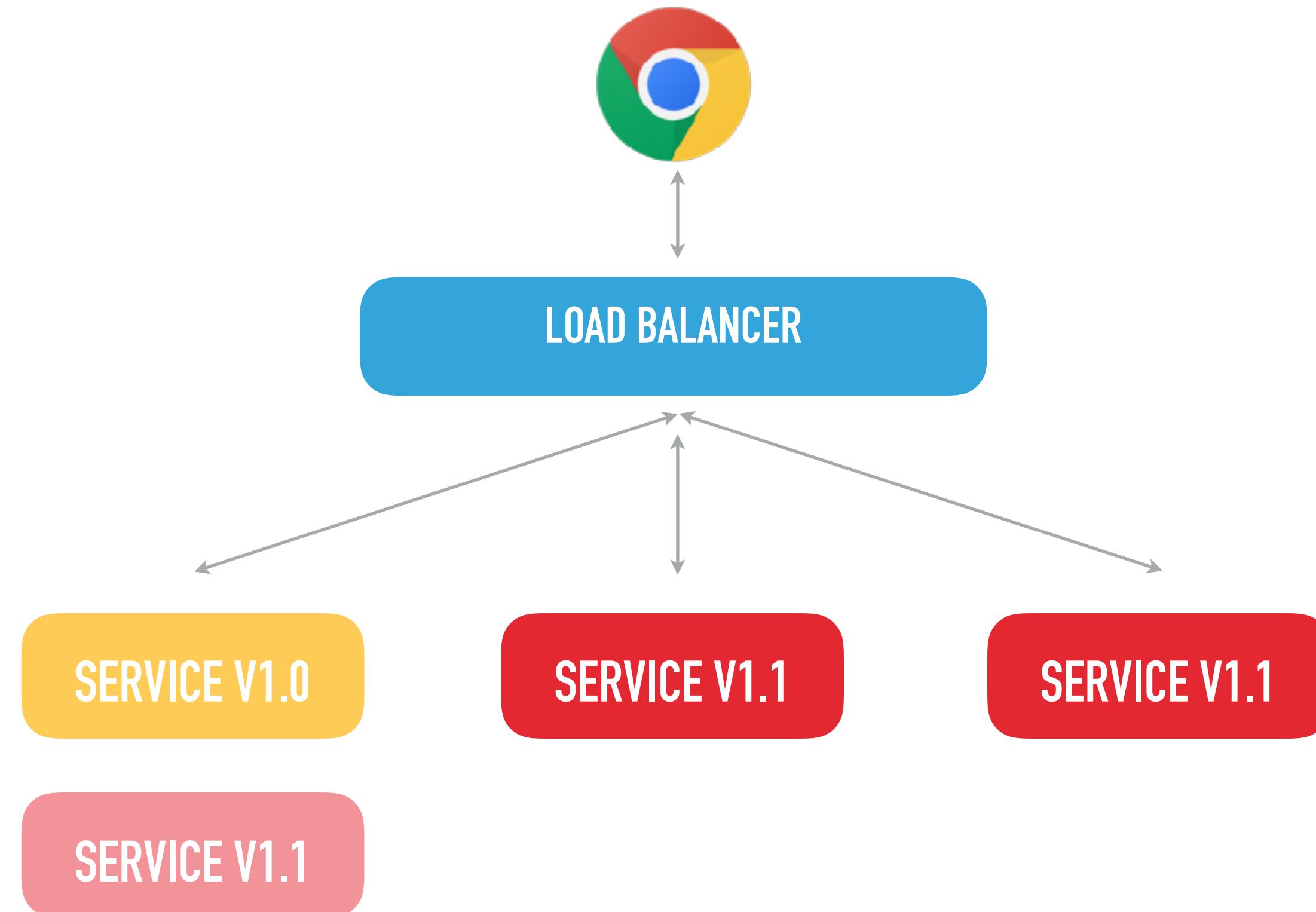
ROLLING UPDATE



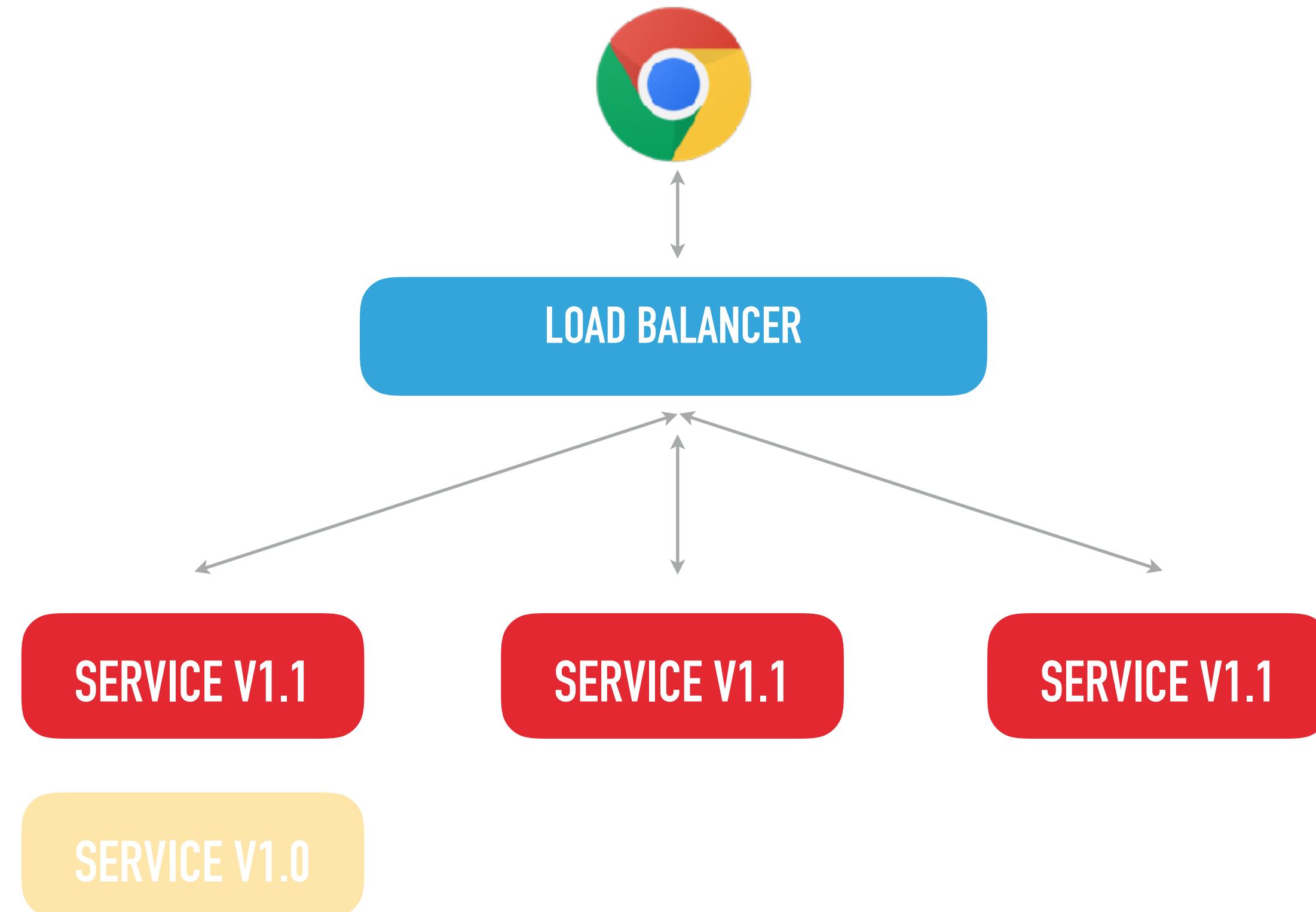
ROLLING UPDATE



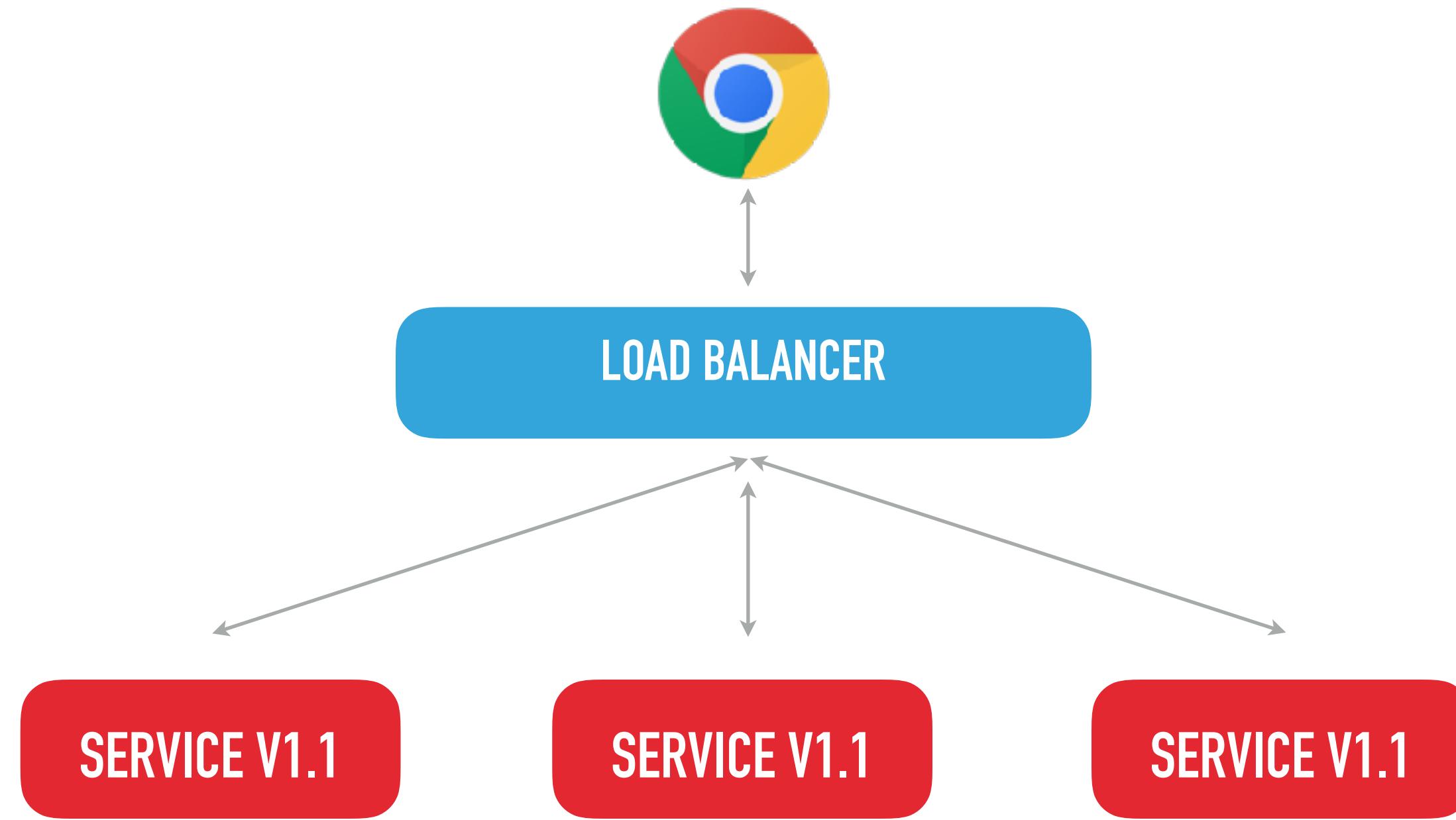
ROLLING UPDATE



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

1087

Pipeline

Changes

Tests

Artifacts



Login



Branch: -

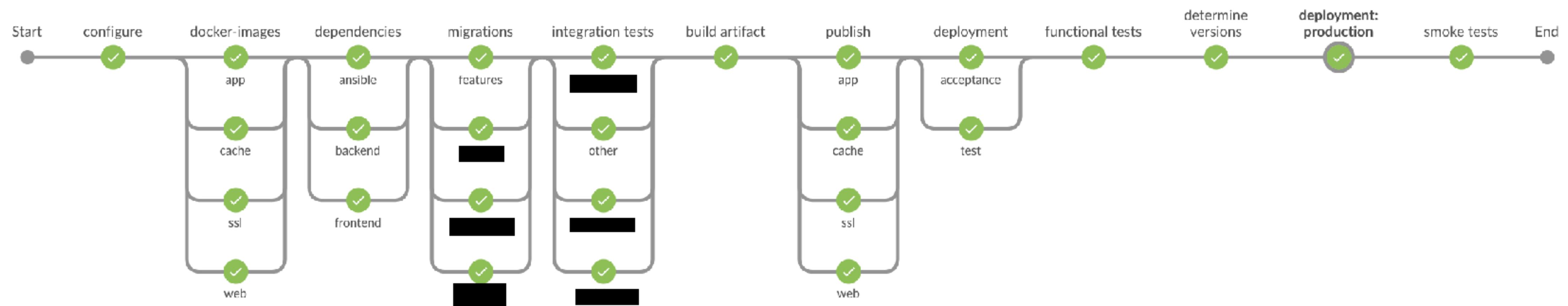
⌚ 21m 13s

Changes by [REDACTED]

Commit: -

⌚ 42 minutes ago

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

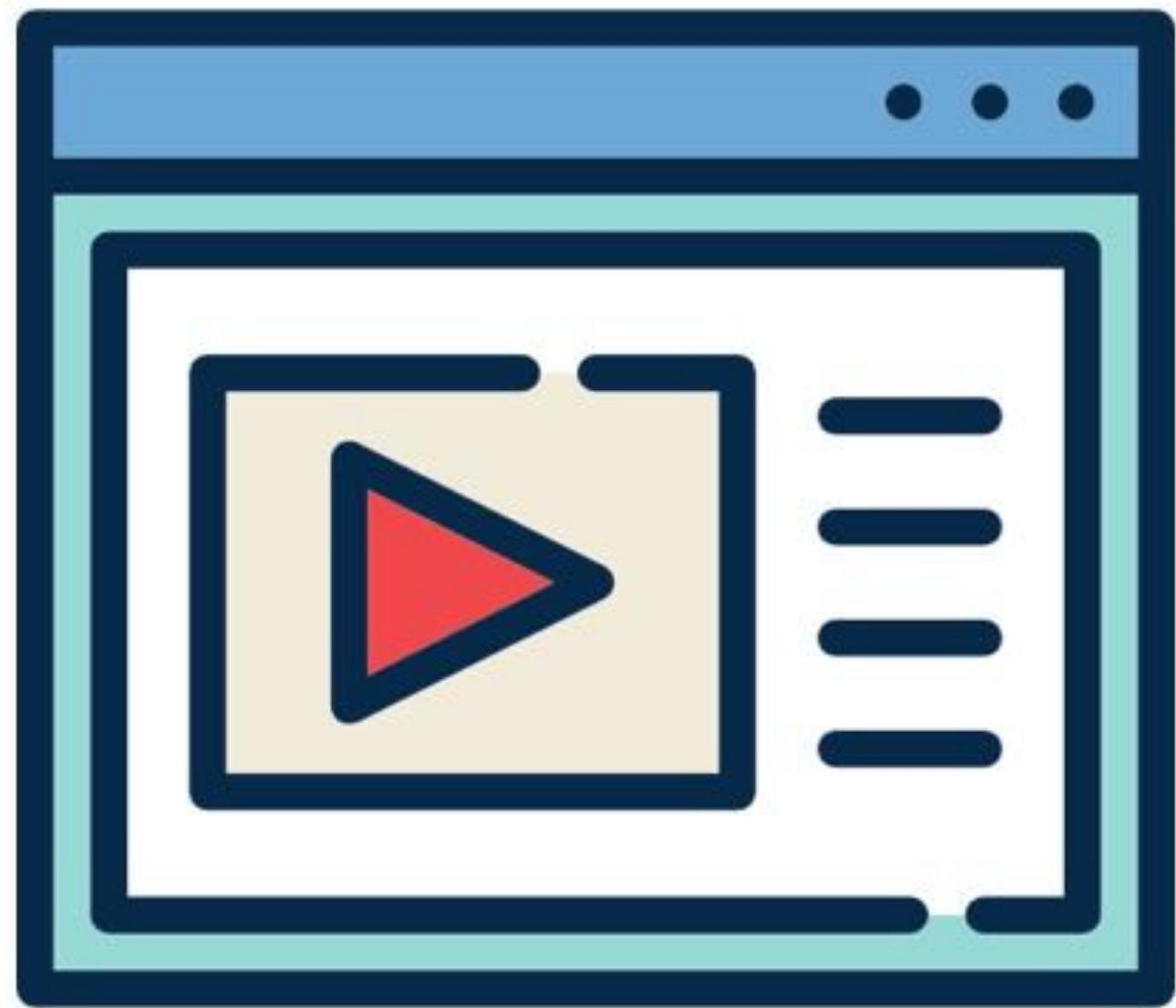
RELEASE TIMING?



FEATURE TOGGLES

**DECUPLE
DEPLOYMENTS
FROM
RELEASES**

New Feature



Feature Flag or Toggle



Consumers

Feature flag dashboard

			Clear all user states
Content	Global state	User state	
:s	OFF ON	ON OFF No override	
aWeb	OFF ON	ON OFF No override	
agementFeature	OFF ON	ON OFF No override	
itFeature	OFF ON	ON OFF No override	
ent	OFF ON	ON OFF No override	
lequestFeature	OFF ON	ON OFF No override	

CAUTION: FEATURE TOGGLE DEBT



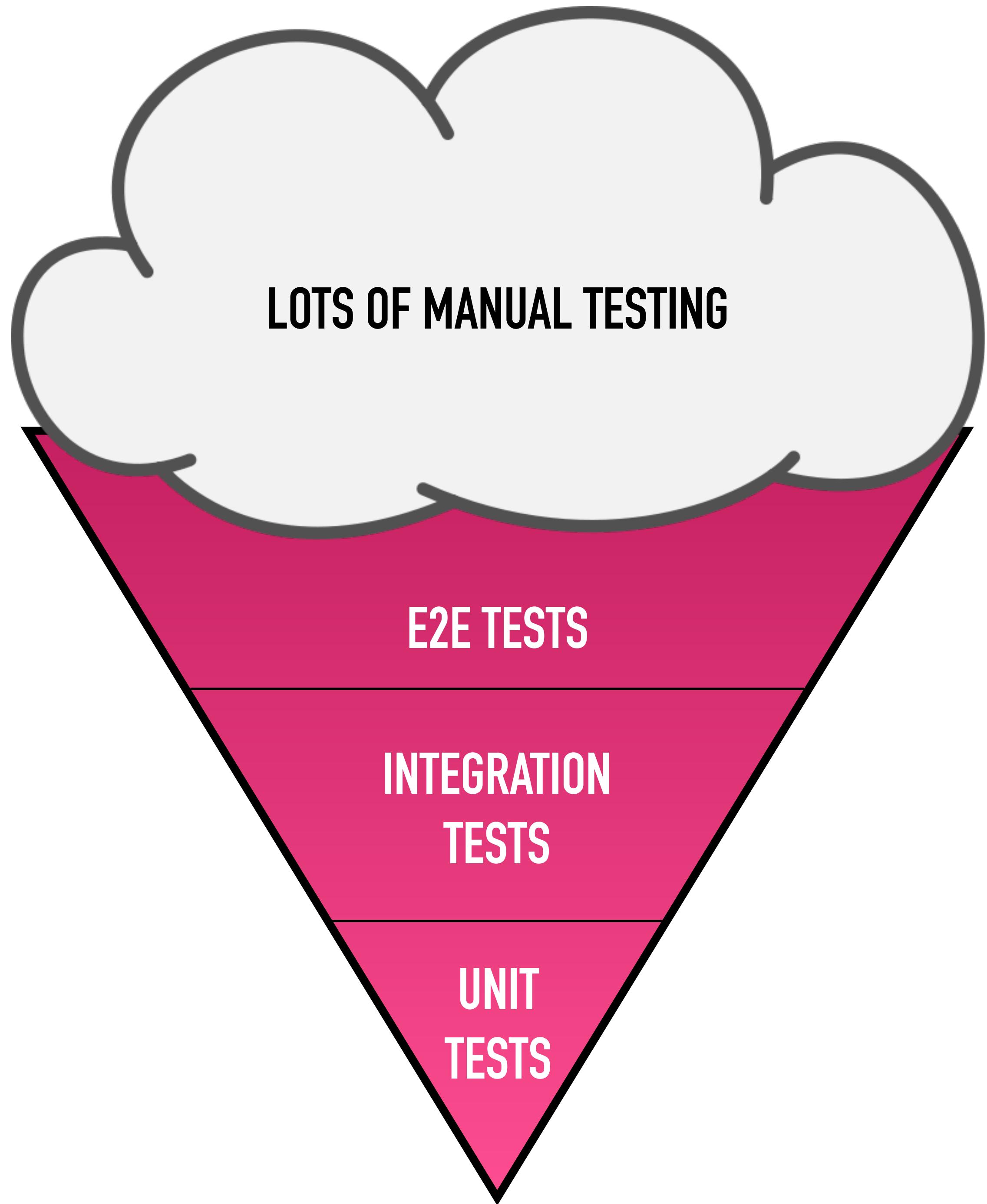
PHASE 3: IMPROVEMENTS & TESTING

PIPELINE SPEED

BUY A BIGGER BOX

CLEVER PIPELINE IMPROVEMENTS: PARALLELIZE

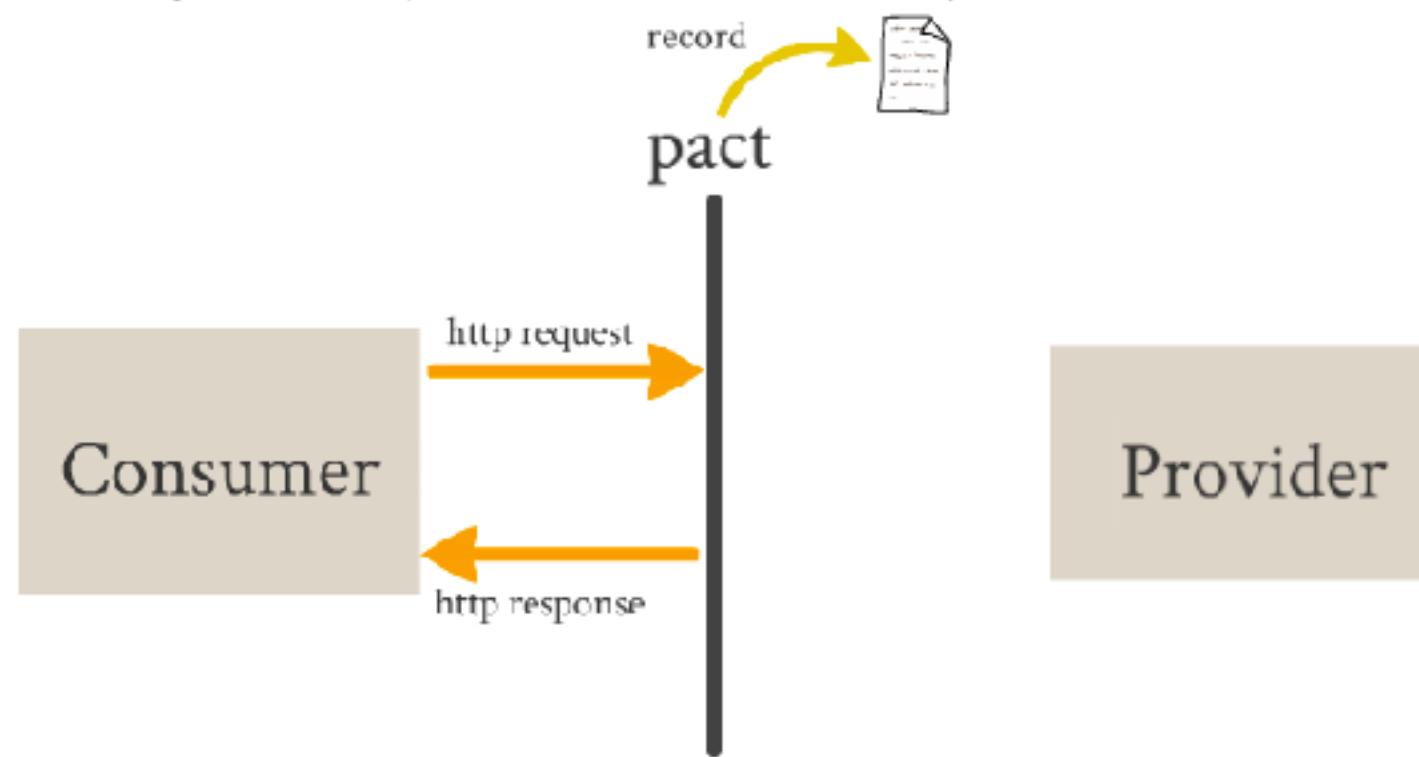
TESTING IMPROVEMENTS



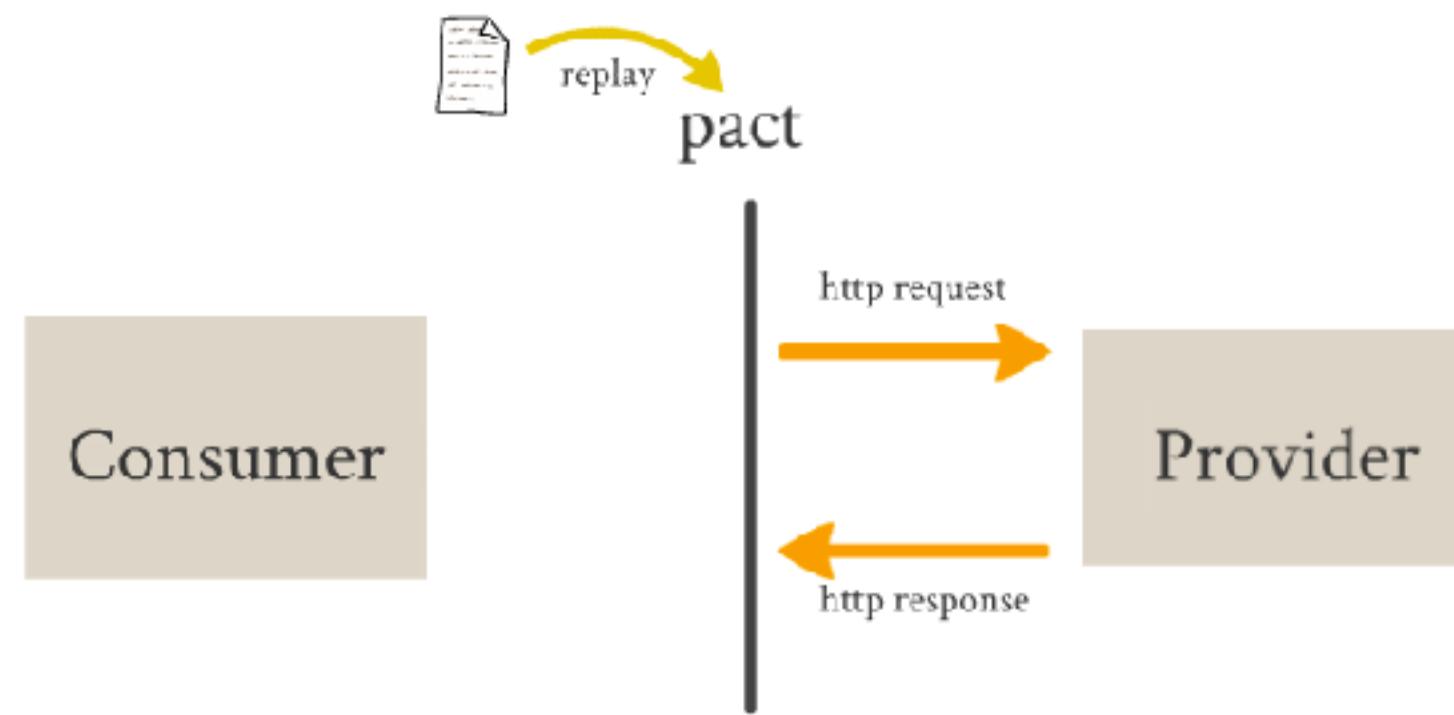
QUALITY ENGINEERING

AUTOMATION

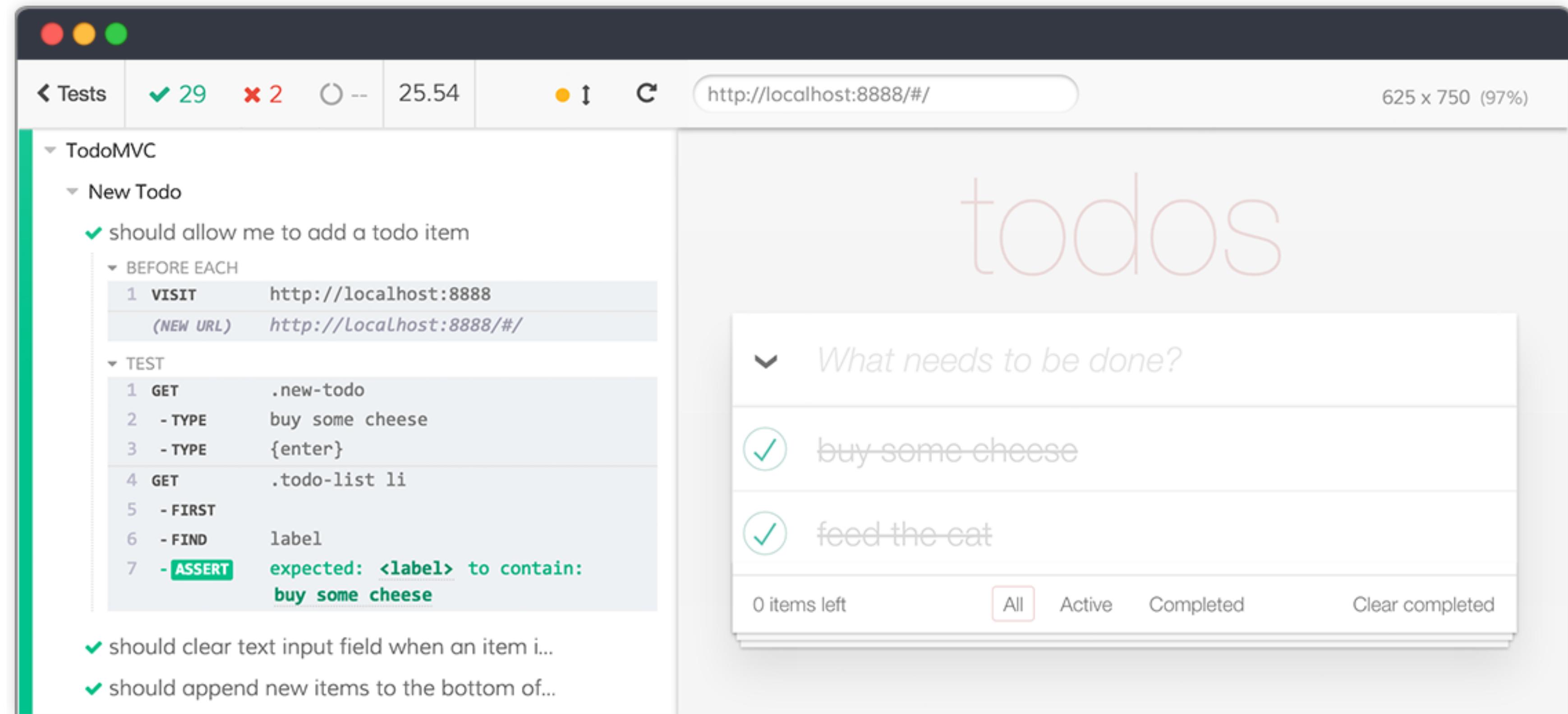
Step 1 - Define Consumer expectations



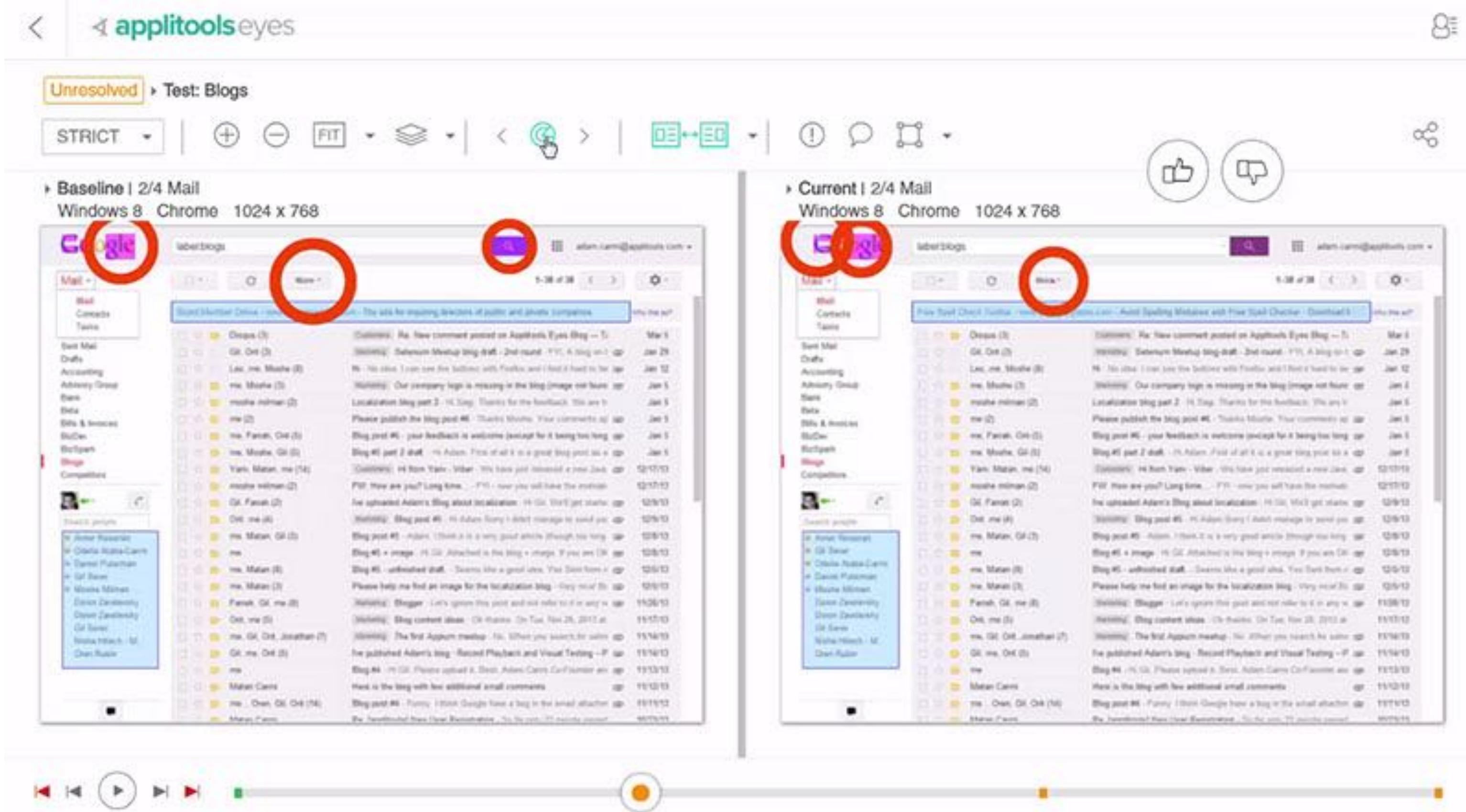
Step 2 - Verify expectations on Provider



CONTRACT TESTING



UI TESTING



VISUAL TESTING

[Security] Bump bower from 1.8.2 to 1.8.8 #80

Merged dependabot merged 1 commit into master from dependabot/npm_and_yarn/bower-1.8.8 7 days ago

Conversation 0 Commits 1 Checks 0 Files changed 1 +2 -2

dependabot bot commented 7 days ago

Bumps bower from 1.8.2 to 1.8.8. This update includes security fixes.

▼ Vulnerabilities fixed
Sourced from *The Node Security Working Group*.

Arbitrary File Write Through Archive Extraction
attackers can write arbitrary files when a malicious archive is extracted.

Affected versions: <1.8.7

► Release notes
► Commits
► Maintainer changes

compatibility 88%

Dependabot will resolve any conflicts with this PR as long as you don't alter it yourself. You can also

Contributor + ...

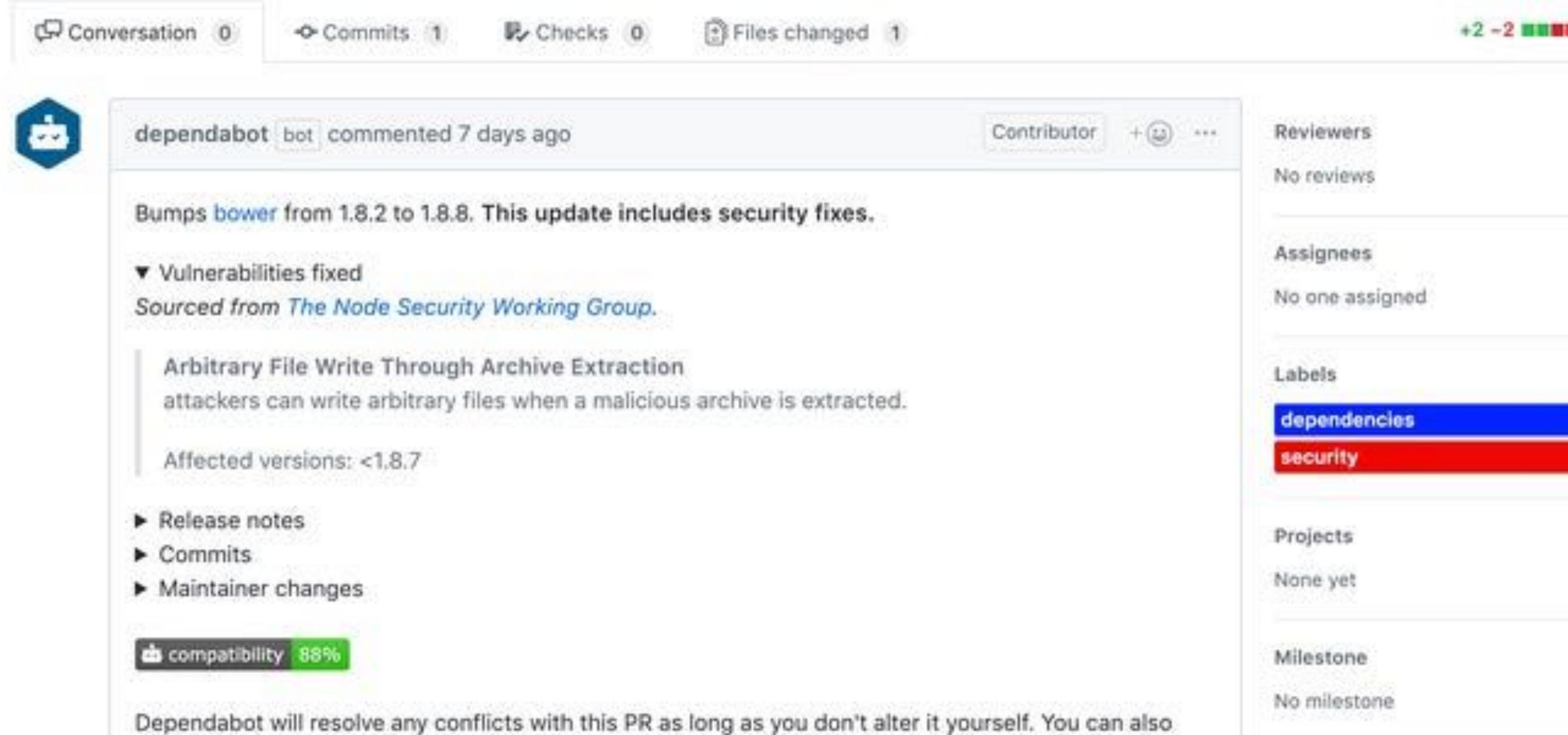
Reviewers
No reviews

Assignees
No one assigned

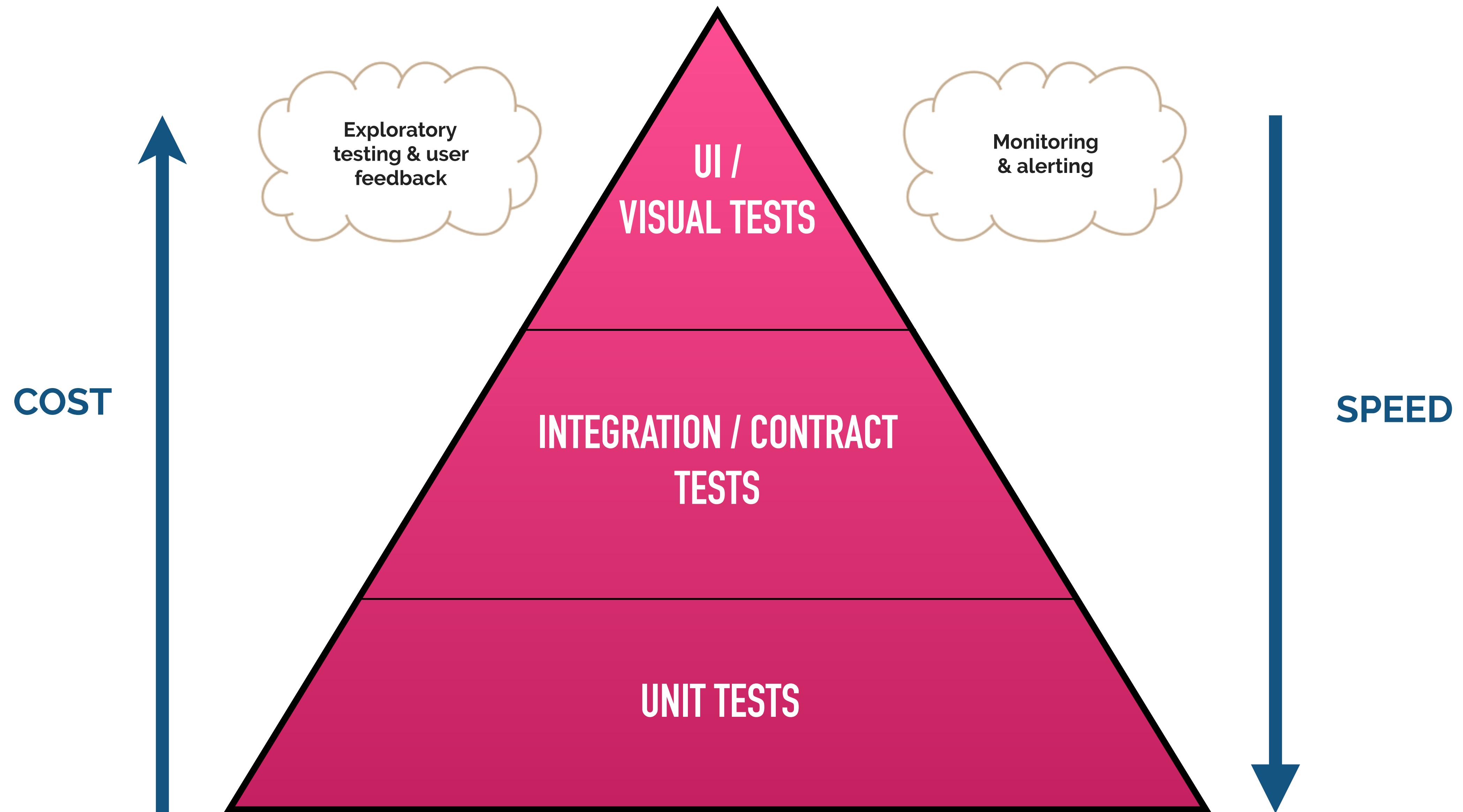
Labels
dependencies
security

Projects
None yet

Milestone
No milestone



DEPENDENCY SCANNING





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