CONTINUOUS SECURITY

IN THE DEVOPS WORLD

JULIEN VEHENT
MOZILLA SECURITY

tip: navigate with left/right arrows

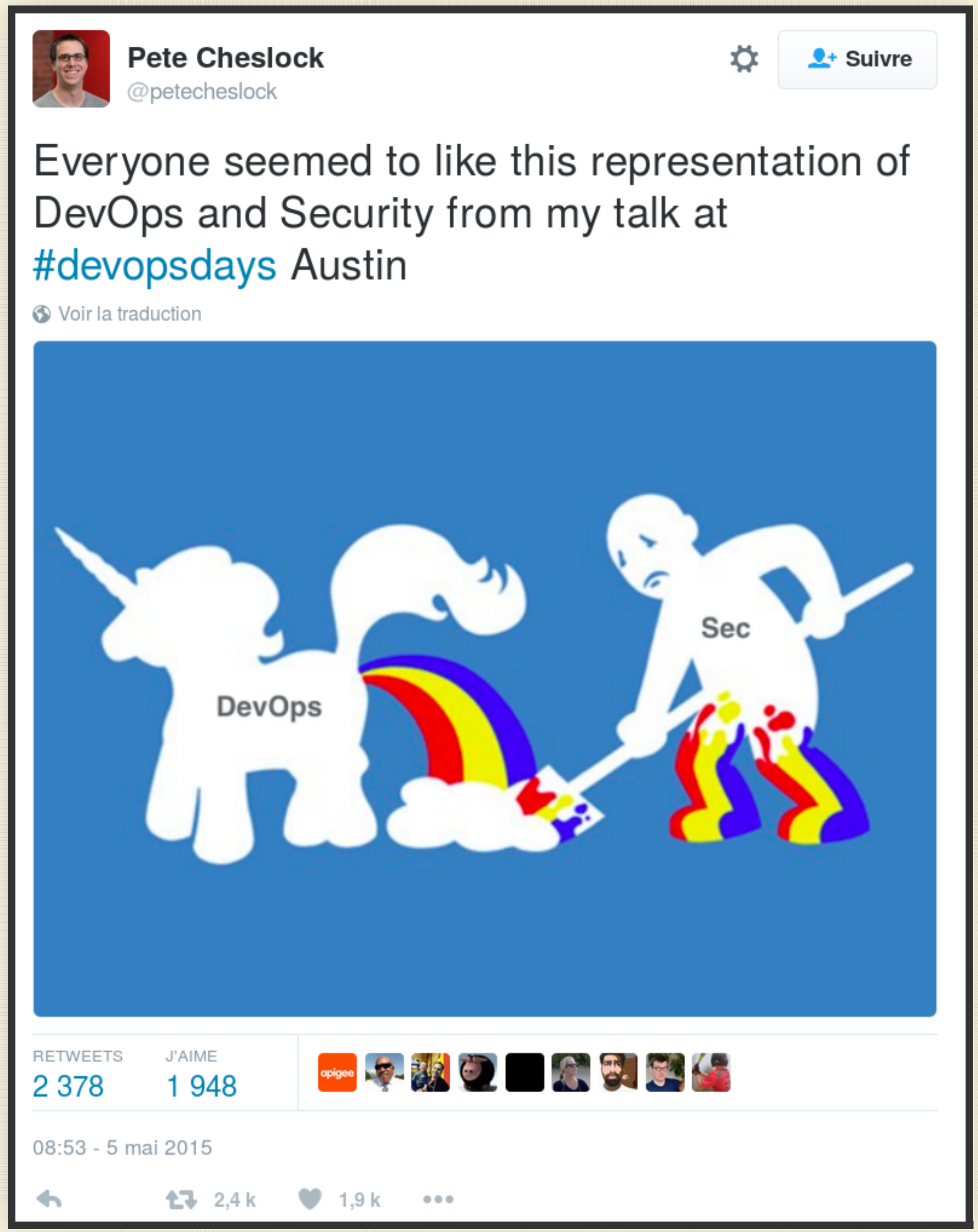
$WHOAMI

- Firefox Services Security Lead
- Infrastructure defense & incident response
- sec tools coder: MIG, sops, TLS Observatory, ...
- 50% ops, 50% dev, 50% security

@jvehent on twitter

THIS TALK IS ABOUT DEVOPS AND SECURITY

IT'S ABOUT AVOIDING THIS



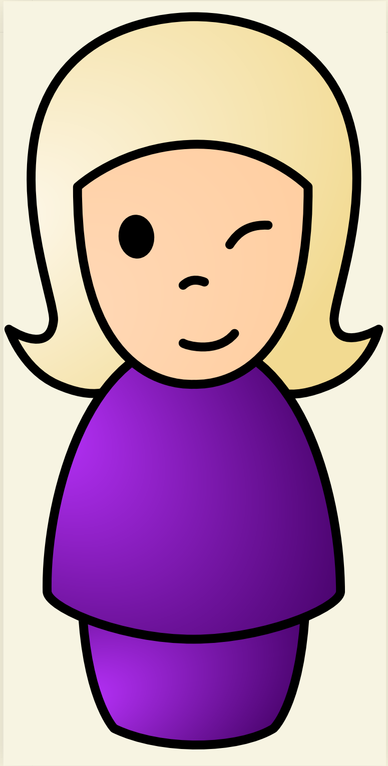
Pete Cheslock
@petecheslock

Everyone seemed to like this representation of DevOps and Security from my talk at #devopsdays Austin

RETWEETS 2 378 J'AIME 1 948

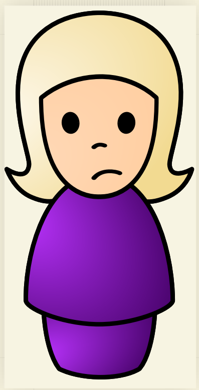
08:53 - 5 mai 2015

MEET SAMANTHA



She's a Full Stack developer

SAM USED TO WORK @SLOWCORP



She didn't like it much

- Internal private repos
- Manual deployment by ops, would take weeks
- Different platform between dev & prod
- No access to cool tools everyone else uses

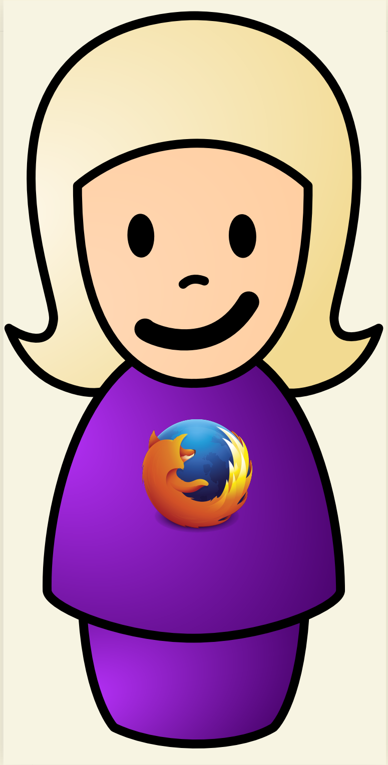
SPEED MATTERS

Traditional ops where deployments take entire weeks aren't acceptable anymore.

To compete, startups need fast release cycles.

15min from patch to prod is the new standard!

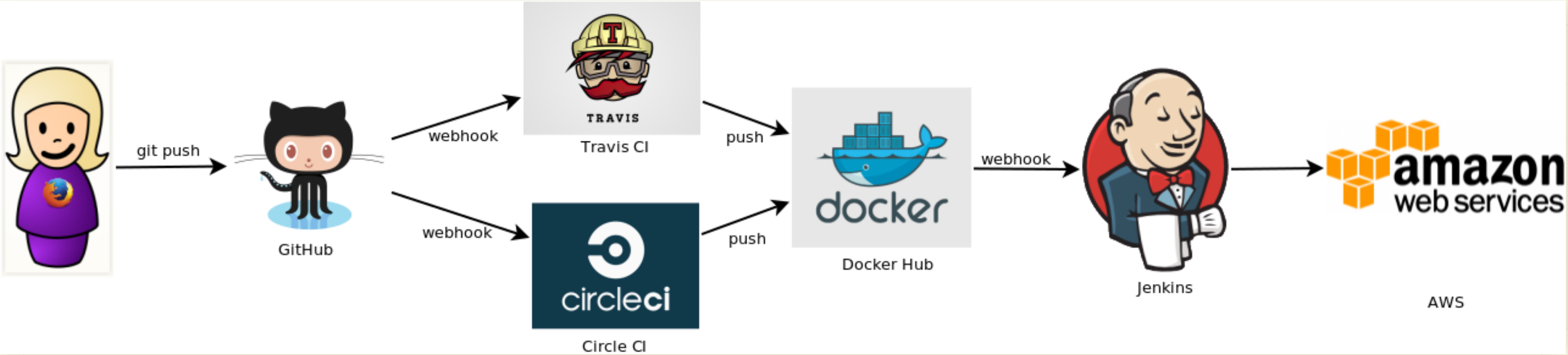
SAM NOW WORKS AT MOZILLA



She gets to use all the cool stuff!

WHAT'S THE COOL STUFF

- Code in public Github repo
- Circle/Travis CI to run tests
- Docker to build and deploy applications
- Continuous Deployment via Jenkins in AWS
- Logs in Kibana, monitoring in Datadog


git push
GitHub
webhook
webhook
TRAVIS
Travis CI
circleci
Circle CI
push
push
docker
Docker Hub
webhook
Jenkins
amazon web services
AWS

IN AN IDEAL WORLD, ALL DEPLOYS ARE AUTOMATED AND INSTANTANEOUS

in the real world, we're not quite there yet, but you get the point

SECURITY VERSUS DEVOPS

AKA. *THE WRONG WAY*

- DevOps team optimizes for fast iterations
- Security team optimizes for fewer incidents

Both sides typically work against each other, actively arming both the roadmap and security of the product

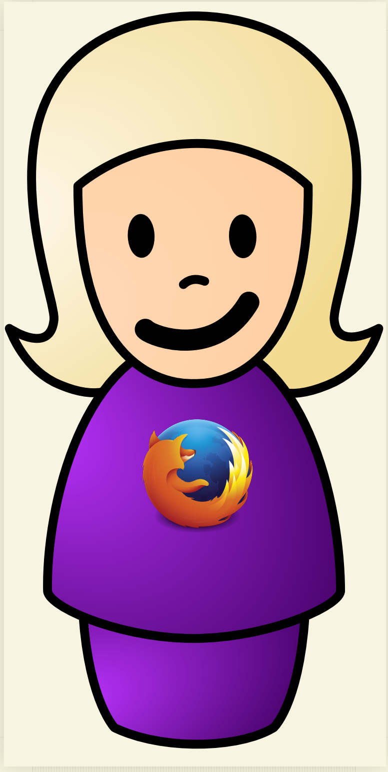
SECURITY INTO DEVOPS

1. Test Driven Security (TDS) integrated into the delivery pipeline. Use security tests to gradual improve application & infrastructure security.
2. Monitoring & blocking attacks, via fraud detection techniques and incident response.
3. Managing risks throughout the life-cycle of the service.

CONTINUOUS SECURITY AT MOZILLA

Walkthrough through the life-cycle of a project, from inception to retirement

SAM IS BUILDING A NEW SERVICE



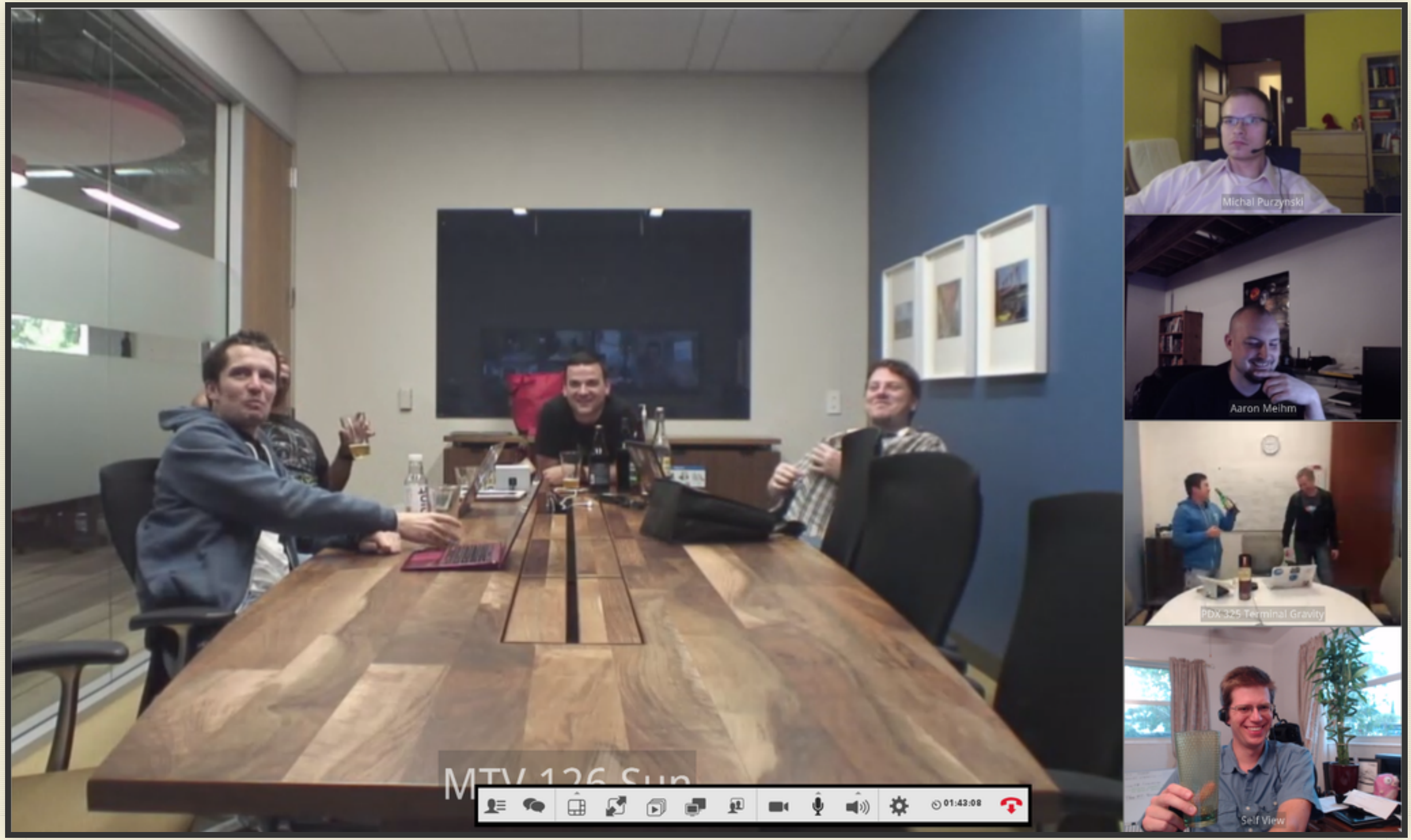
CuteFox: a REST API that sends webpush notifications to Firefox users with photos of cute foxes.

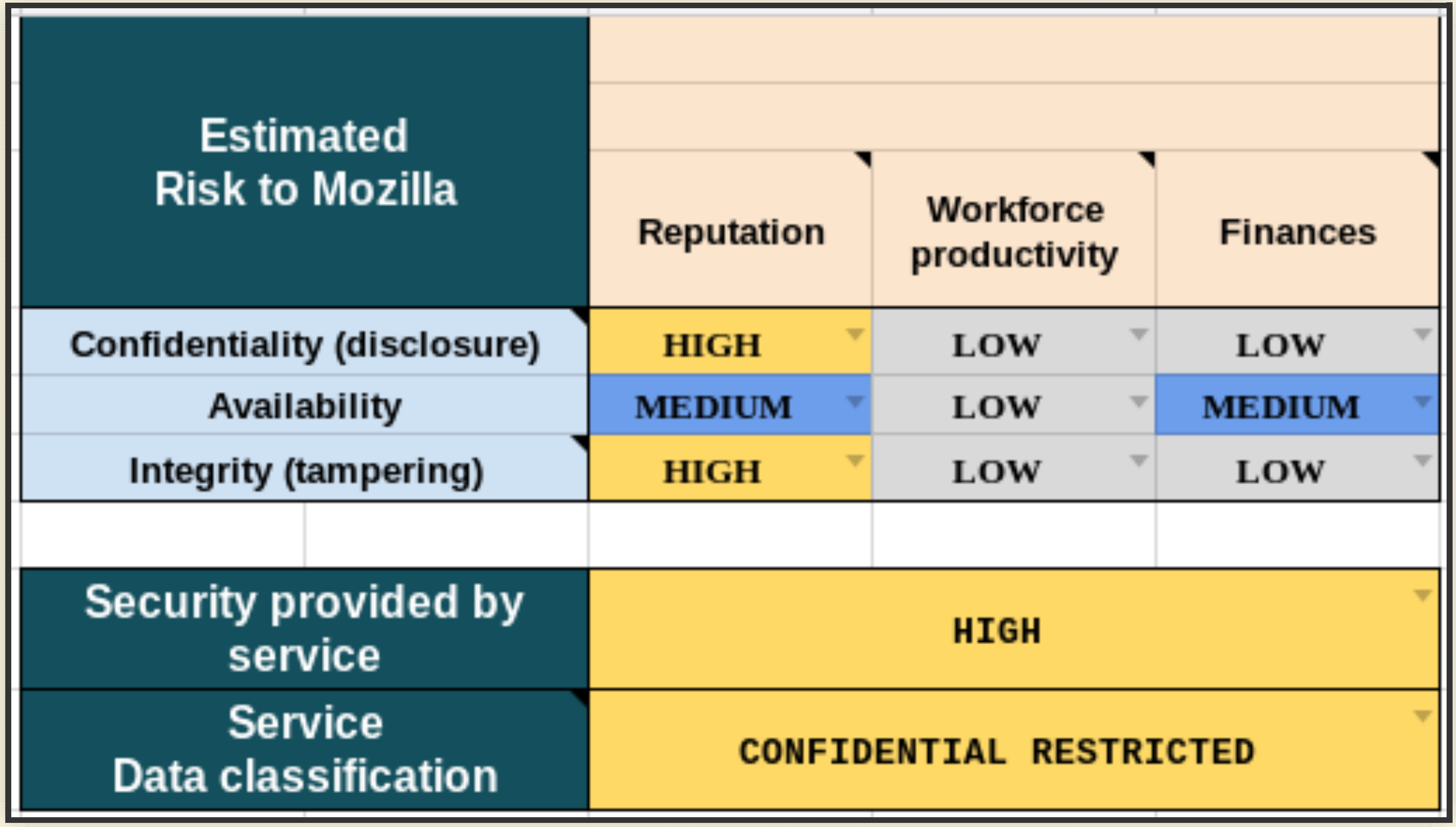
WHEN THE PROJECT STARTS, WE TALK RISK TOGETHER

RRA: RAPID RISK ASSESSMENT

A ~30min **friendly** discussion between the devs, ops, products managers and security team to go over the business risks of the project

DONE REMOTELY!





Estimated Risk to Mozilla	Reputation	Workforce productivity	Finances
Confidentiality (disclosure)	HIGH	LOW	LOW
Availability	MEDIUM	LOW	MEDIUM
Integrity (tampering)	HIGH	LOW	LOW

Security provided by service	HIGH
Service Data classification	CONFIDENTIAL RESTRICTED

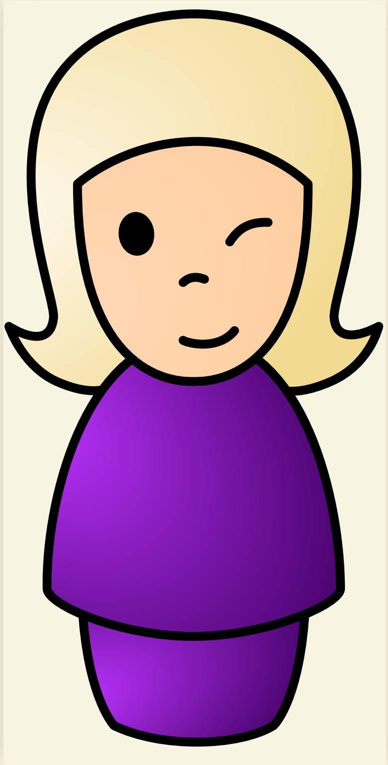
A risk summary table from the RRA

RRA OUTPUTS RECOMMENDATIONS

We capture those recommendation into a "Risk Summary" bug. The bug stays open for the lifetime of the service and serves as a tracker for security discussions related to the project

THE PROJECT TEAM UNDERSTANDS THE RISKS THEIR PROJECT IS EXPOSED TO.

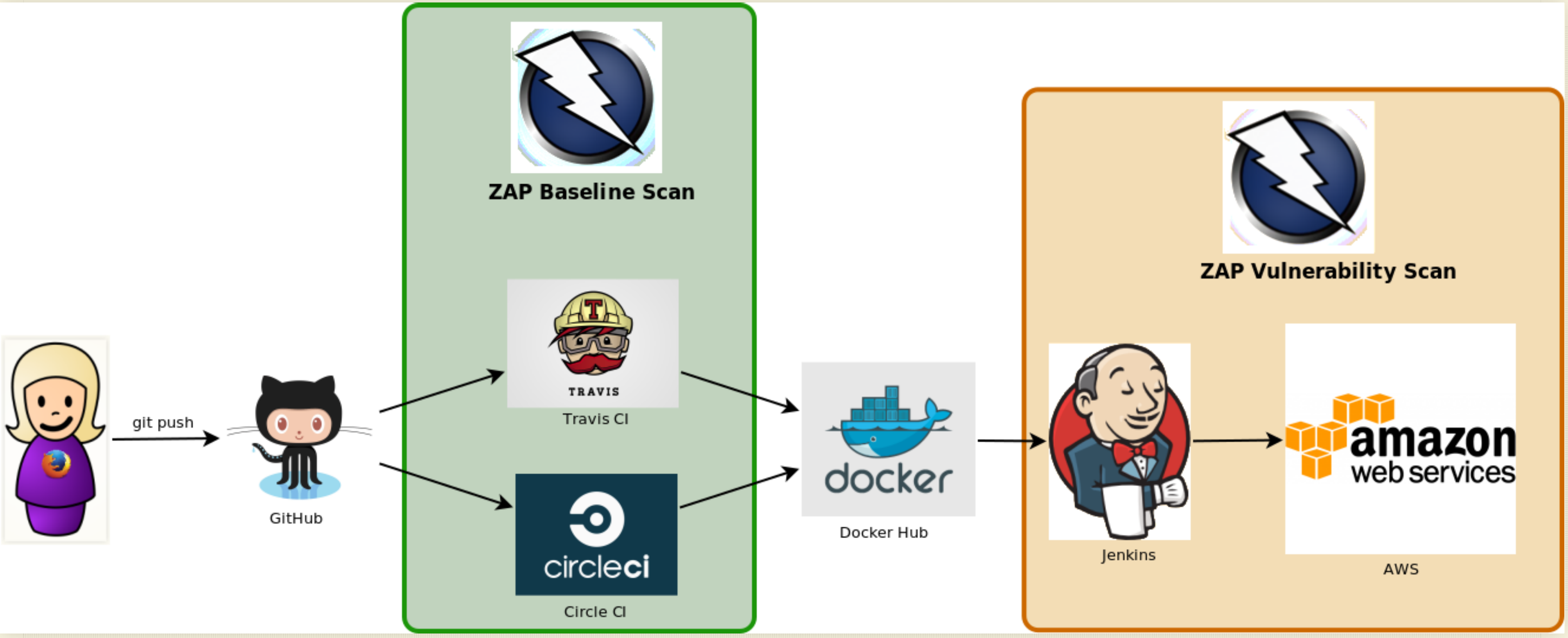
SAM GOES CODING



WE HELP SAM AVOID COMMON WEBAPP VULNERABILITIES

- Mozilla Web Security Guidelines wiki.mozilla.org/Security/Guidelines/Web_Security
- OWASP ZAP Scanning github.com/zaproxy/ZAP-Baseline-Scan
- Require baseline security on all websites (CSP, Secure Cookies, TLS Only, ...)

TEST DRIVEN SECURITY FOR WEB APPLICATIONS



ZAP EXAMPLE IN CIRCLECI

```
test:
  override:
    - docker run mozilla/cutefox &

    # pull down the ZAP docker container
    - docker pull owasp/zap2docker-weekly

    # Run ZAP against the application
    - >
        docker run -t owasp/zap2docker-weekly zap-baseline.py
        -t http://172.17.0.2:8080/

    # Shut down the application container
    - >
        docker kill
        $(docker ps |grep mozilla/cutefox
        | awk '{print $1}')
```

PASS/FAIL OUTPUT, LIKE UNIT TESTS

```
PASS: Absence of Anti-CSRF Tokens [40014]

WARN: Web Browser XSS Protection Not Enabled [10016] x 3
    http://172.17.0.2:8080/
    http://172.17.0.2:8080//robots.txt
    http://172.17.0.2:8080//sitemap.xml
```

TEST DRIVEN SECURITY

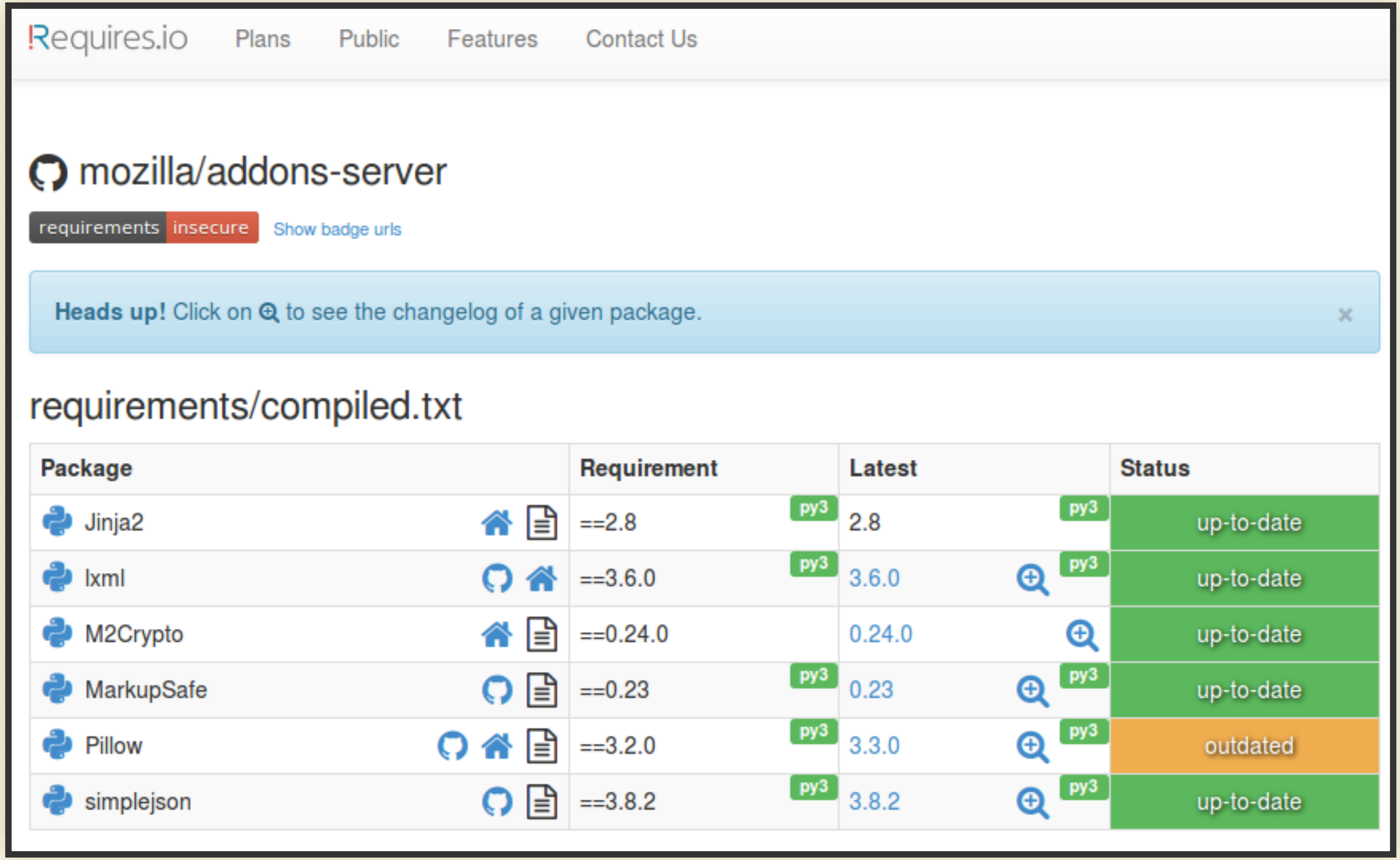
Similar to TDD: Write the security tests first, let them fail, implement the security control then verify the tests pass

- Security team writes the tests
- Developers implement the controls

WE ALSO ASK SAM TO KEEP HER APP UP TO DATE

- Node.JS: NSP, Greenkeeper.io
- Python: requires.io, pip --outdated
- Go: govend

TDS FOR DEPENDENCY MANAGEMENT



mozilla/addons-server

Heads up! Click on 🔍 to see the changelog of a given package.

requirements/compiled.txt

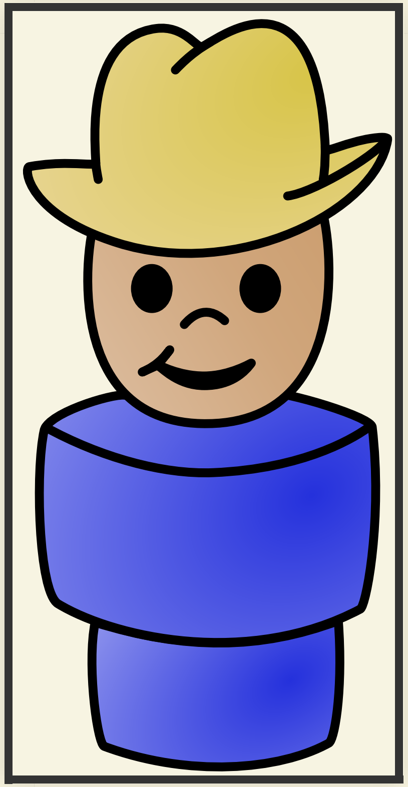
Package	Requirement	Latest	Status
Jinja2	==2.8 py3	2.8 py3	up-to-date
lxml	==3.6.0 py3	3.6.0 py3	up-to-date
M2Crypto	==0.24.0	0.24.0	up-to-date
MarkupSafe	==0.23 py3	0.23 py3	up-to-date
Pillow	==3.2.0 py3	3.3.0 py3	outdated
simplejson	==3.8.2 py3	3.8.2 py3	up-to-date

DEVELOPERS OWN THE OPERATIONAL SECURITY OF THEIR APPLICATION

We don't bolt it on top with WAFs and so on, we build security into the app directly

THEN WE DEPLOY

MEET MAX



He's the Ops guy

MAX HAS TO WRITE ALL THE PROVISIONING CODE

- Build the AWS infra via cloudformation
- Setup the jenkins pipeline to for continuous deployment (Docker container deployed to EC2 instances with Jenkins, Ansible, Cloudformation and Puppet).
- He often helps the devs make architecture decisions, like how to use CDNs, caching, etc...

ONE DOES NOT SIMPLY
SERVE TRAFFIC TO 300M USERS
imgflip.com

WE HELP MAX WITH TOOLS...

- Managing secrets (SOPS) to prevent leaks
- Configuring good TLS on endpoints (TLS Observatory)
- Disabling users that have left the company (Userplex)
- Building crypto services so services don't have to manage keys (Autograph)

AND GUIDELINES

- Require that admin panel must be placed behind VPN
- Perform audits and incident response training with the teams

etc...

SEC TEAM BUILDS SOLUTIONS TO HELP DEVOPS

1. Dev or Ops come see us with a problem
2. We discuss it together
3. Sec or Dev team builds a solution that solve the issue
4. We generalize it so other teams can benefit as well

EXAMPLE: STORING SECRETS IN GIT

Problem: secrets in cleartext files have a bad tendency to leak

Solution: SOPS - encrypt all credentials, decrypt at provisioning

```
# The secrets below are unreadable without access to one of the sops mast
myapp1: ENC[AES256_GCM,data:QsGJGjvQOpoVCIlrYTcOQEfQzriw,iv:ShmgdRNV6UrO
app2:
    db:
        user: ENC[AES256_GCM,data:Arbb,iv:7bjm4ZaVFlxNk3O4M1P67TqfFtXTOH
        password: ENC[AES256_GCM,data:9/jSxNCq0A==,iv:5mk+GS016hKGj6gVfQ
```

TEST DRIVEN SECURITY FOR THE INFRASTRUCTURE

- Test the TLS configuration daily (certificate, ciphersuites, ...)
- [future] Test security groups with mozilla/build-fwunit
- [future] Test AWS IAM policies

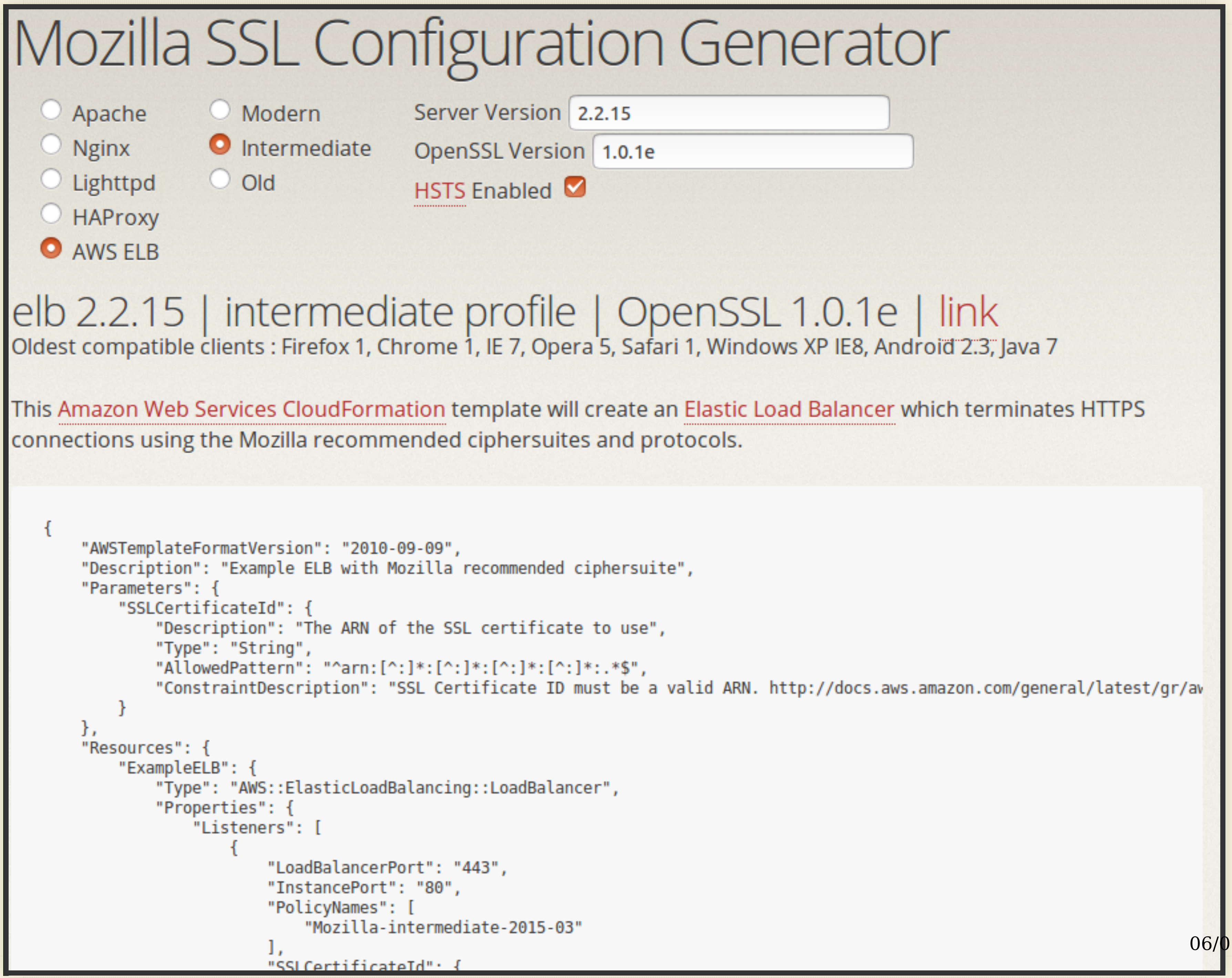
EXAMPLE: TESTING TLS CONFIGURATION

```
$ tlsobs addons.mozilla.org

[...]

--- Analyzers ---
* Mozilla evaluation: intermediate
  - for modern level: remove ciphersuites ECDHE-RSA-AES128-SHA, ECDHE-RSA
  - for modern level: consider adding ciphers ECDHE-ECDSA-AES256-GCM-SHA
  - for modern level: remove protocols TLSv1, TLSv1.1
  - for modern level: consider enabling OCSP stapling
  - for modern level: use a certificate of type ecdsa, not RSA
  - oldest clients: Firefox 1, Chrome 1, IE 7, Opera 5, Safari 1, Windows
```

WHEN TLS CONFIG TEST FAILS, WE DIRECT OPS TO THE CONFIG GENERATOR



Mozilla SSL Configuration Generator

○ Apache ○ Modern Server Version 2.2.15
○ Nginx ● Intermediate OpenSSL Version 1.0.1e
○ Lighttpd ○ Old HSTS Enabled ☑
○ HAProxy
● AWS ELB

elb 2.2.15 | intermediate profile | OpenSSL 1.0.1e | link

Oldest compatible clients : Firefox 1, Chrome 1, IE 7, Opera 5, Safari 1, Windows XP IE8, Android 2.3, Java 7

This Amazon Web Services CloudFormation template will create an Elastic Load Balancer which terminates HTTPS connections using the Mozilla recommended ciphersuites and protocols.

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Example ELB with Mozilla recommended ciphersuite",
    "Parameters": {
        "SSLCertificateId": {
            "Description": "The ARN of the SSL certificate to use",
            "Type": "String",
            "AllowedPattern": "^arn:[^:]*:[^:]*:[^:]*:[^:]*:.*$",
            "ConstraintDescription": "SSL Certificate ID must be a valid ARN. http://docs.aws.amazon.com/general/latest/gr/aw
        }
    },
    "Resources": {
        "ExampleELB": {
            "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
            "Properties": {
                "Listeners": [
                    {
                        "LoadBalancerPort": "443",
                        "InstancePort": "80",
                        "PolicyNames": [
                            "Mozilla-intermediate-2015-03"
                        ],
                        "SSLCertificateId": {
```

IT'S LAUNCH DAY! FOXES EVERYWHERE!

UNTIL BAD GUYS START ATTACKING CUTEFOX

INCIDENT RESPONSE

NO ONE IN THE DEVOPS TEAM SLEEPS UNTIL THE FIRE IS OUT

INCIDENTS SUCK

but they are great for

- Team building: Nothing like going through hell together to build trust!
- Roadmaps: Incidents **always** bump up the priority of security features.
- Security maturity: no amount of testing compares to an incident to evaluate the reliability of a service.

CONTINUOUS SECURITY IS A CYCLE

1. design new feature
2. assess risks
3. implement feature
4. test security
5. deploy
6. get attacked
7. fight back
8. learn
9. rinse and repeat

SECURITY MUST BE PART OF THE PRODUCT

Not an afterthought built on top

- Be a member of the DevOps team
- Understand the roadmap
- Share the successes
- Share the failures
- Write code that makes things better

It's not SecDevOps, it's just DevOps.
Security is a natural component of it.

THANK YOU

jvehent.github.io/continuous-security-talk