safe advisory.

Protecting pipelines

Secure software delivery using
the OWASP CI/CD Top 10

Thank You to Our Sponsors and Hosts!

OWASP NEW ZEALAND owasp.org.nz

AppSec NZ appsec.org.nz

AUT UNIVERSITY TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

DEFEND™

fastly®

QUANTUM SECURITY

DATACOM

aura INFORMATION SECURITY POWERED BY KORDIA

IriusRisk

snyk

dta Defence Technology Agency

planit an NRI company

CyberCX

tesserent

FIGHTING FOR FAIR FOR KIWI BUSINESS

safeadvisory.

safeadvisory.

Without them, this Conference couldn't happen.

Who were you again?

- Reformed Sysadmin
- Who fell into dev projects
- Was accidentally at birthplace of CI/CD
- Did startups for a while (OK, 10 years)
- Now a security consultant at …

hello@safeadvisory.co.nz

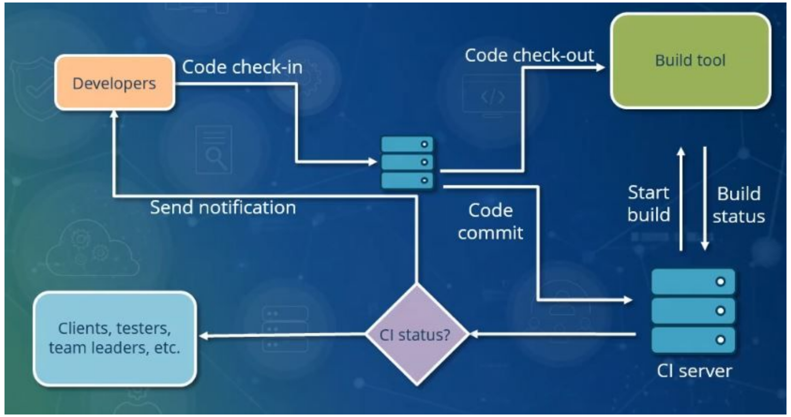
safeadvisory.

www.safeadvisory.co.nz

What’s this talk about?

- A guided tour through the OWASP CI/CD Top 10
- Your tour guide’s thoughts on CI/CD more broadly
- Oh, and roundabouts

Let’s travel back in time…



Continuous **integration**

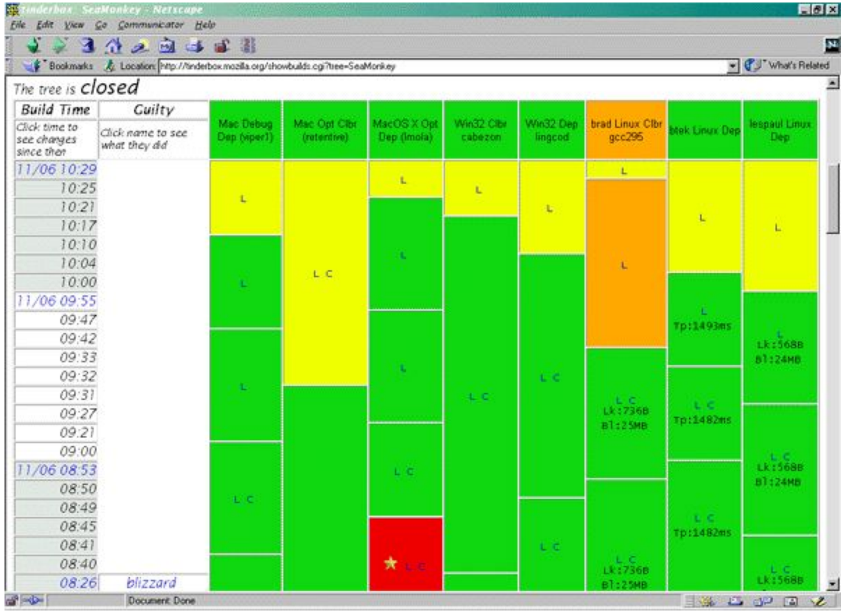
1990s

- Branching is an antipattern
- No tools needed, just an extra workstation
- Of course we wrote tools …

Mozilla Tinderbox

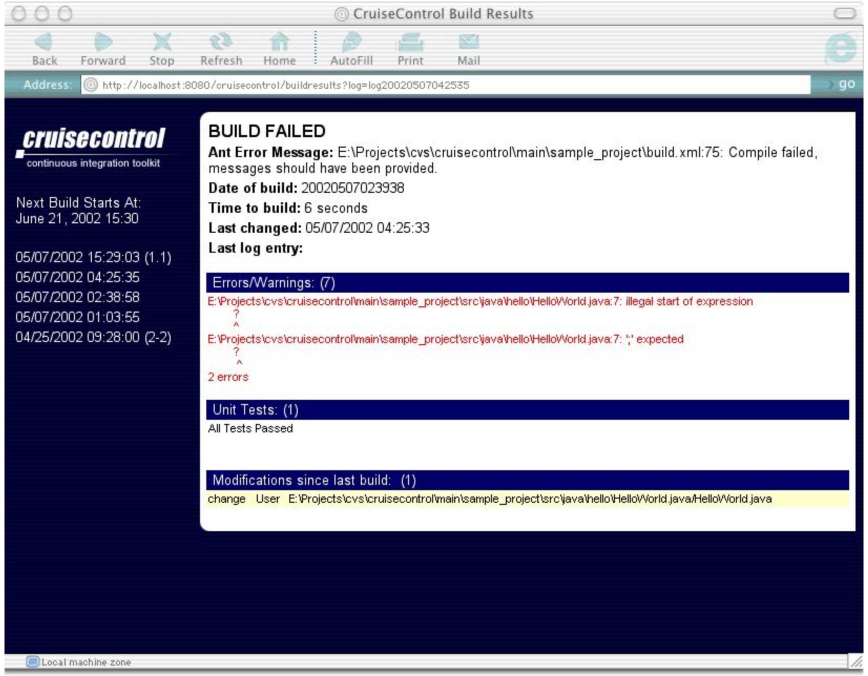
1998

The first CI tool that wasn't cron jobs and spit


Tinderbox SeaMonkey - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://tinderbox.mozilla.org/showbuilds.cgi?tree=SeaMonkey
What's Related
Document: Done

The tree is closed

Build Time	Guilty
Click time to see changes since then	Click name to see what they did
11/06 10:29	
10:25	
10:21	
10:17	
10:10	
10:04	
10:00	
11/06 09:55	
09:47	
09:42	
09:33	
09:32	
09:31	
09:27	
09:21	
09:00	
11/06 08:53	
08:50	
08:49	
08:45	
08:41	
08:40	
08:26	blizzard



http://localhost:8080/cruisecontrol/buildresults?log=log20020507042535

cruisecontrol
continuous integration toolkit

Next Build Starts At:
June 21, 2002 15:30

05/07/2002 15:29:03 (1.1)
05/07/2002 04:25:35
05/07/2002 02:38:58
05/07/2002 01:03:55
04/25/2002 09:28:00 (2-2)

BUILD FAILED
Ant Error Message: E:\Projects\cvs\cruisecontrol\main\sample_project\build.xml:75: Compile failed, messages should have been provided.
Date of build: 20020507023938
Time to build: 6 seconds
Last changed: 05/07/2002 04:25:33
Last log entry:

Errors/Warnings: (7)

```
E:\Projects\cvs\cruisecontrol\main\sample_project\src\java\hello\HelloWorld.java:7: illegal start of expression
       ?
       ^
E:\Projects\cvs\cruisecontrol\main\sample_project\src\java\hello\HelloWorld.java:7: ';' expected
       ?
        ^
2 errors
```

Unit Tests: (1)
All Tests Passed

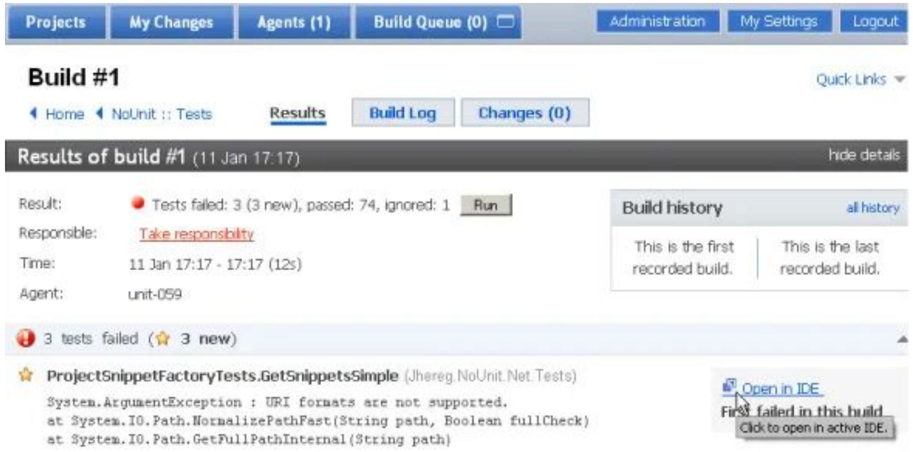
Modifications since last build: (1)
change User E:\Projects\cvs\cruisecontrol\main\sample_project\src\java\hello\HelloWorld.java/HelloWorld.java

CruiseControl 2002

Who remembers CVS?

safe advisory.

TeamCity
2006



Build #1

Results of build #1 (11 Jan 17:17)

Result:	Tests failed: 3 (3 new), passed: 74, ignored: 1
Responsible:	Take responsibility
Time:	11 Jan 17:17 - 17:17 (12s)
Agent:	unit-059

Build history

This is the first recorded build. This is the last recorded build.

3 tests failed (3 new)

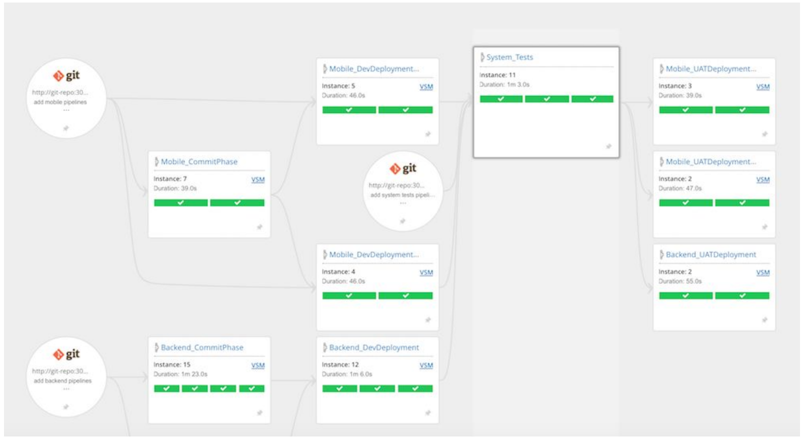
ProjectSnippetFactoryTests.GetSnippetsSimple (Jhereg.NoUnit.Net.Tests)

```
System.ArgumentException : URI formats are not supported.
at System.IO.Path.NormalizePathFast(String path, Boolean fullCheck)
at System.IO.Path.GetFullPathInternal(String path)
```

First failed in this build

safe advisory.

That’s CI, but what is CI/CD?



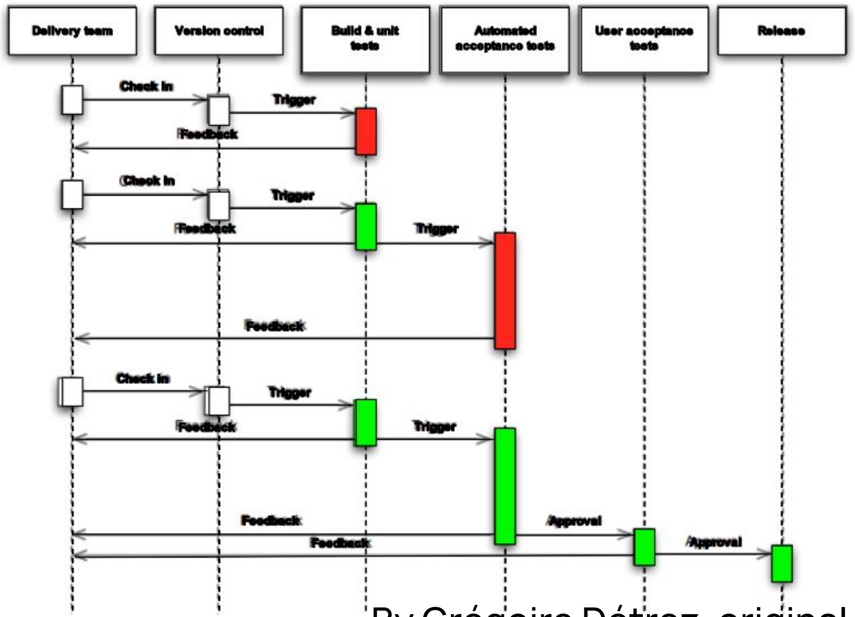
GoCD/Cruise
2007

One of the first products to attempt building from source to production.

safe advisory.

"At an abstract level, **a deployment pipeline is an automated manifestation of your process for getting software from version control into the hands of your users**. Every change to your software goes through a complex process on its way to being released. That process involves **building the software, followed by the progress of these builds through multiple stages of testing and deployment**."

Jez Humble and David Farley
Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation



Continuous delivery 2010s

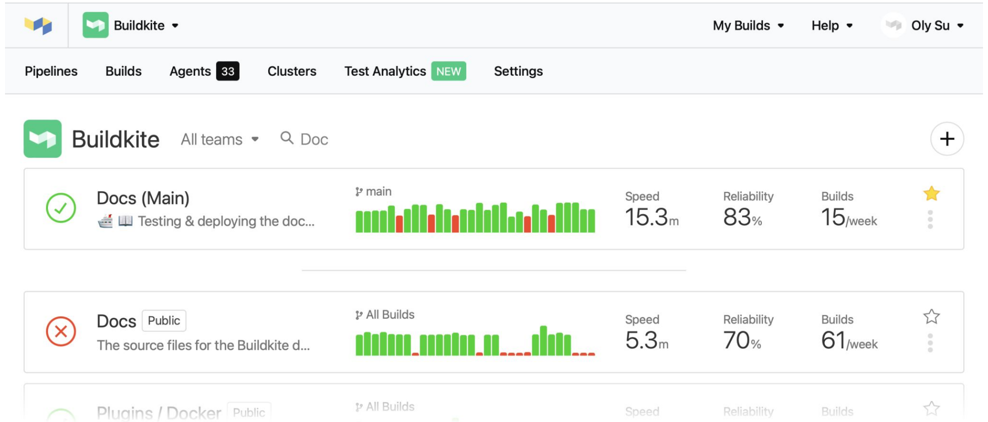
Let's take builds all the way to production

Continuous Delivery book described a lot of techniques in use on projects around the world

CI/CD

(really just Continuous Delivery)

Buildkite
2013-



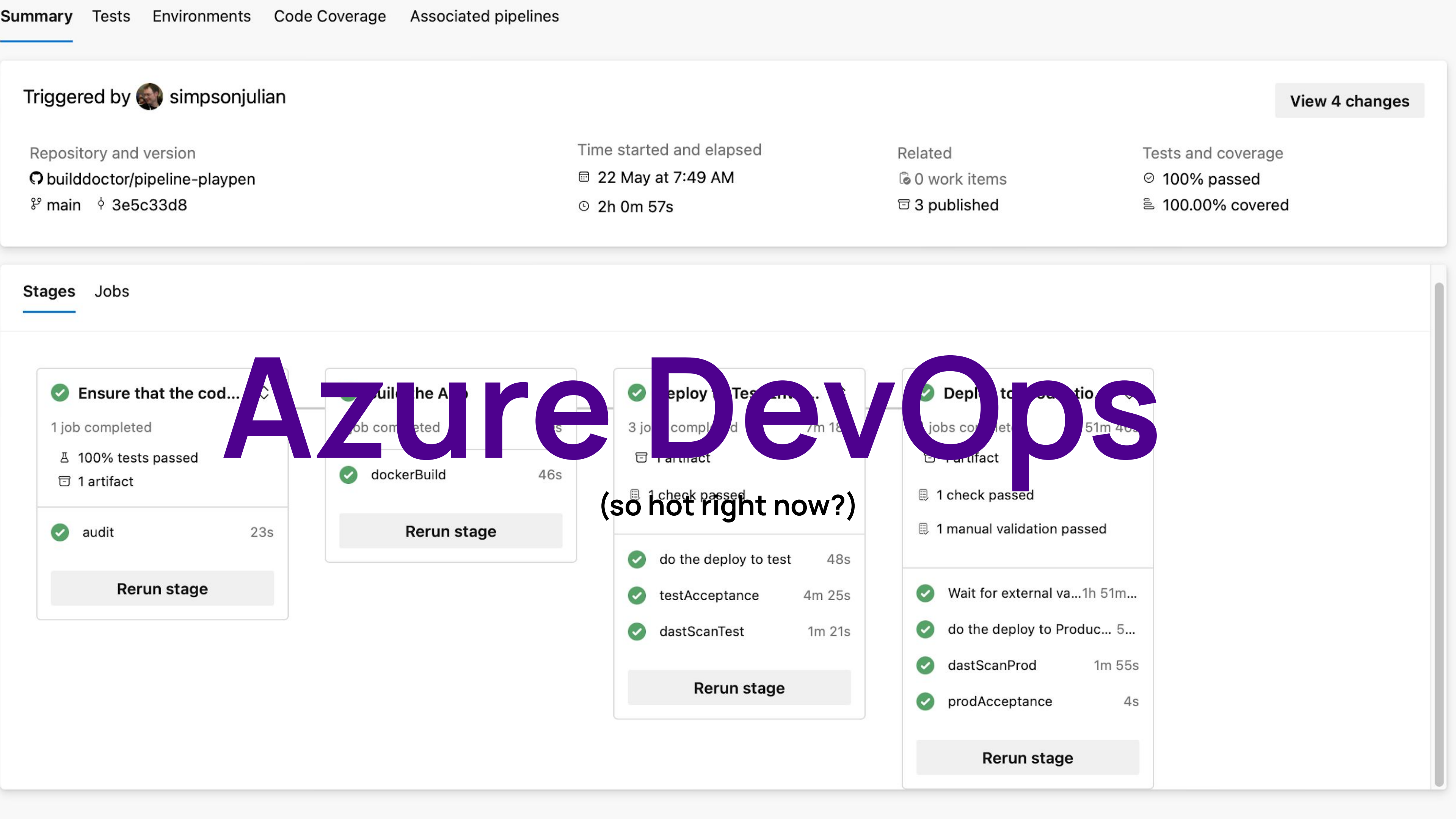
Docs (Main) — Testing & deploying the doc... — main — Speed 15.3m — Reliability 83% — Builds 15/week

Docs Public — The source files for the Buildkite d... — All Builds — Speed 5.3m — Reliability 70% — Builds 61/week

Plugins / Docker Public — All Builds — Speed — Reliability — Builds

Azure DevOps

(so hot right now?)



Triggered by simpsonjulian

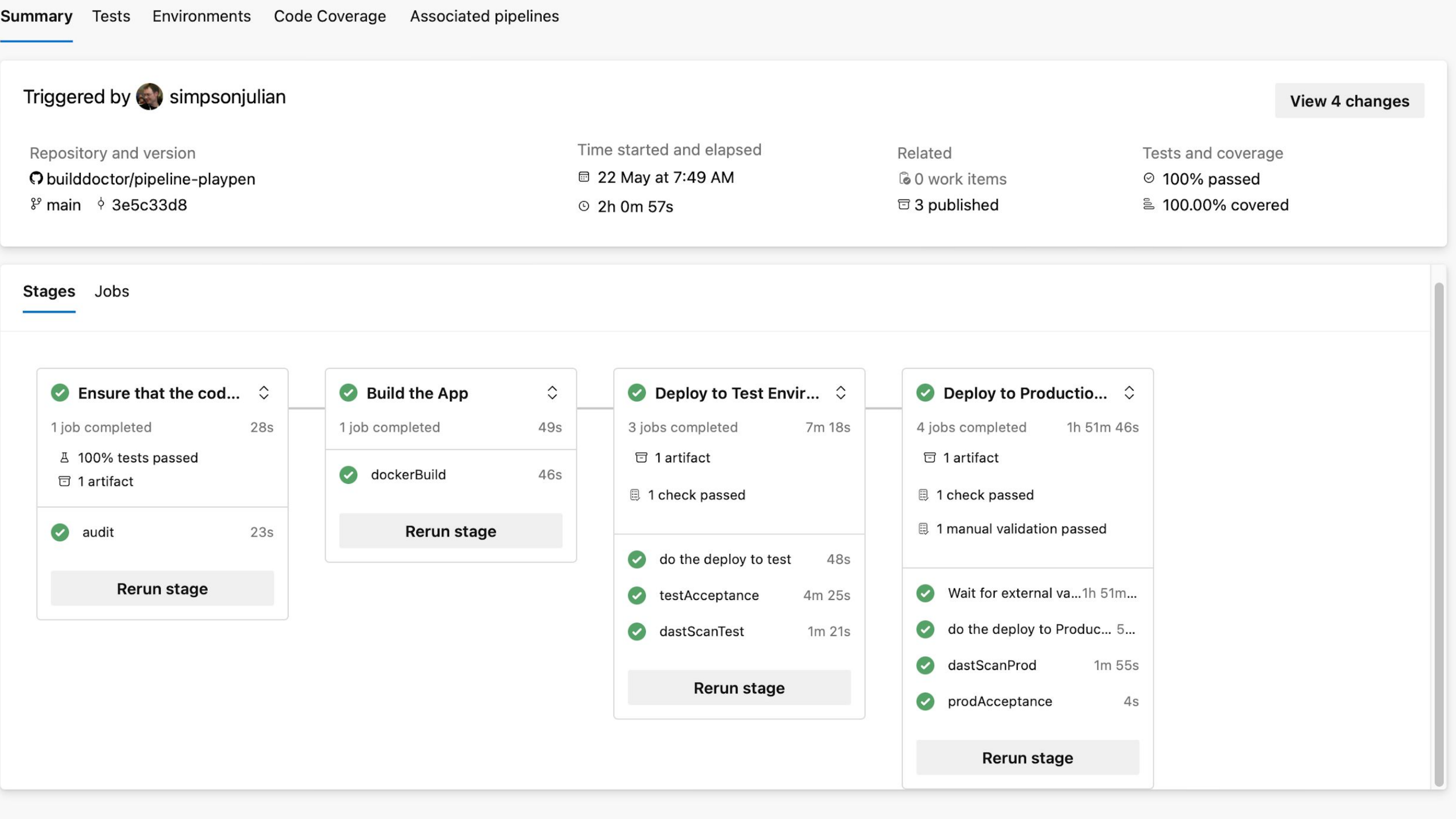
Repository and version	Time started and elapsed	Related	Tests and coverage
builddoctor/pipeline-playpen	22 May at 7:49 AM	0 work items	100% passed
main 3e5c33d8	2h 0m 57s	3 published	100.00% covered

Ensure that the cod...
1 job completed
100% tests passed
1 artifact
audit 23s

dockerBuild 46s

1 check passed
do the deploy to test 48s
testAcceptance 4m 25s
dastScanTest 1m 21s

1 check passed
1 manual validation passed
Wait for external va... 1h 51m...
do the deploy to Produc... 5...
dastScanProd 1m 55s
prodAcceptance 4s



Triggered by simpsonjulian

View 4 changes

Repository and version
builddoctor/pipeline-playpen
main 3e5c33d8

Time started and elapsed
22 May at 7:49 AM
2h 0m 57s

Related
0 work items
3 published

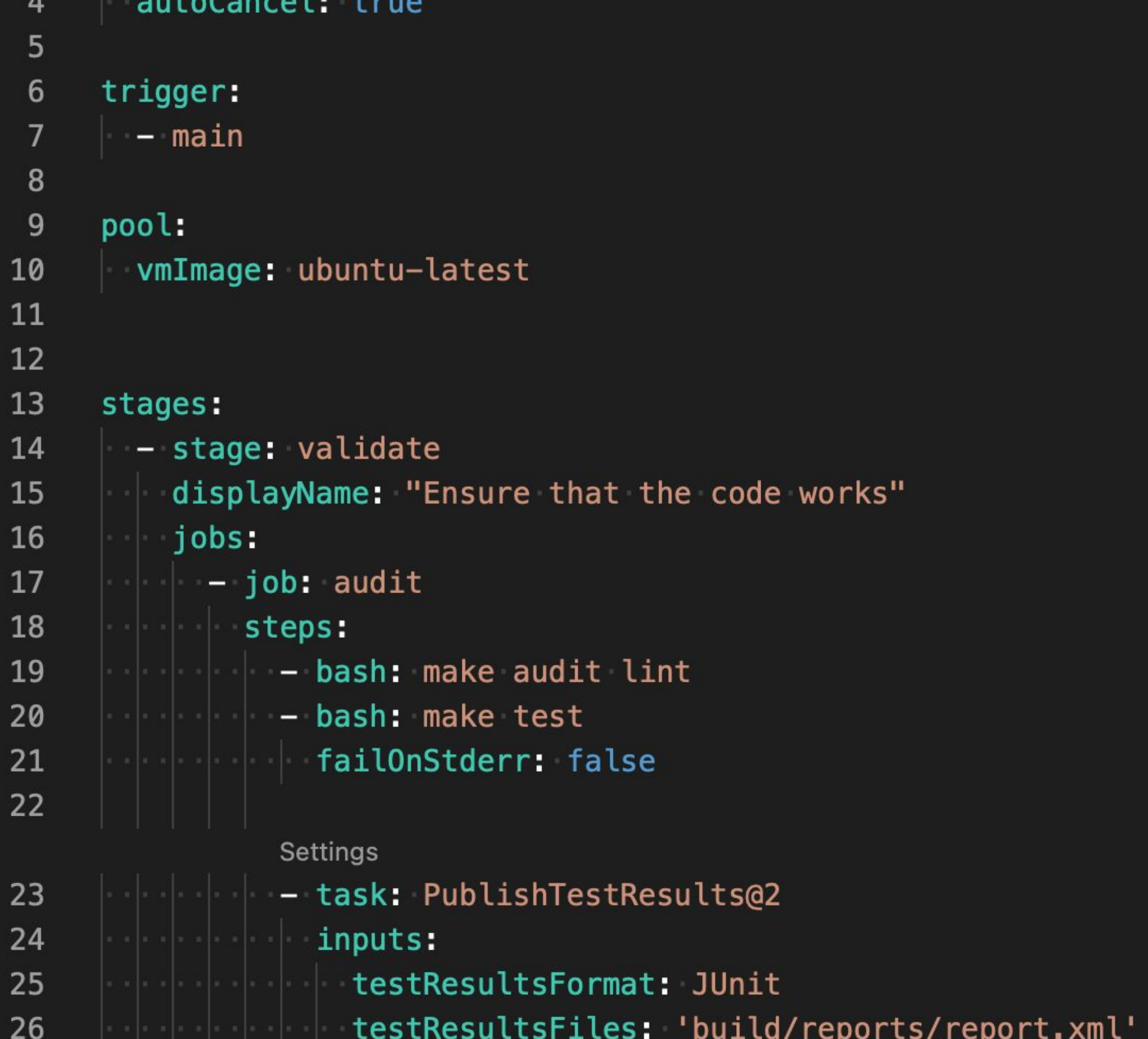
Tests and coverage
100% passed
100.00% covered

Ensure that the cod...
1 job completed 28s
100% tests passed
1 artifact
audit 23s
Rerun stage

Build the App
1 job completed 49s
dockerBuild 46s
Rerun stage

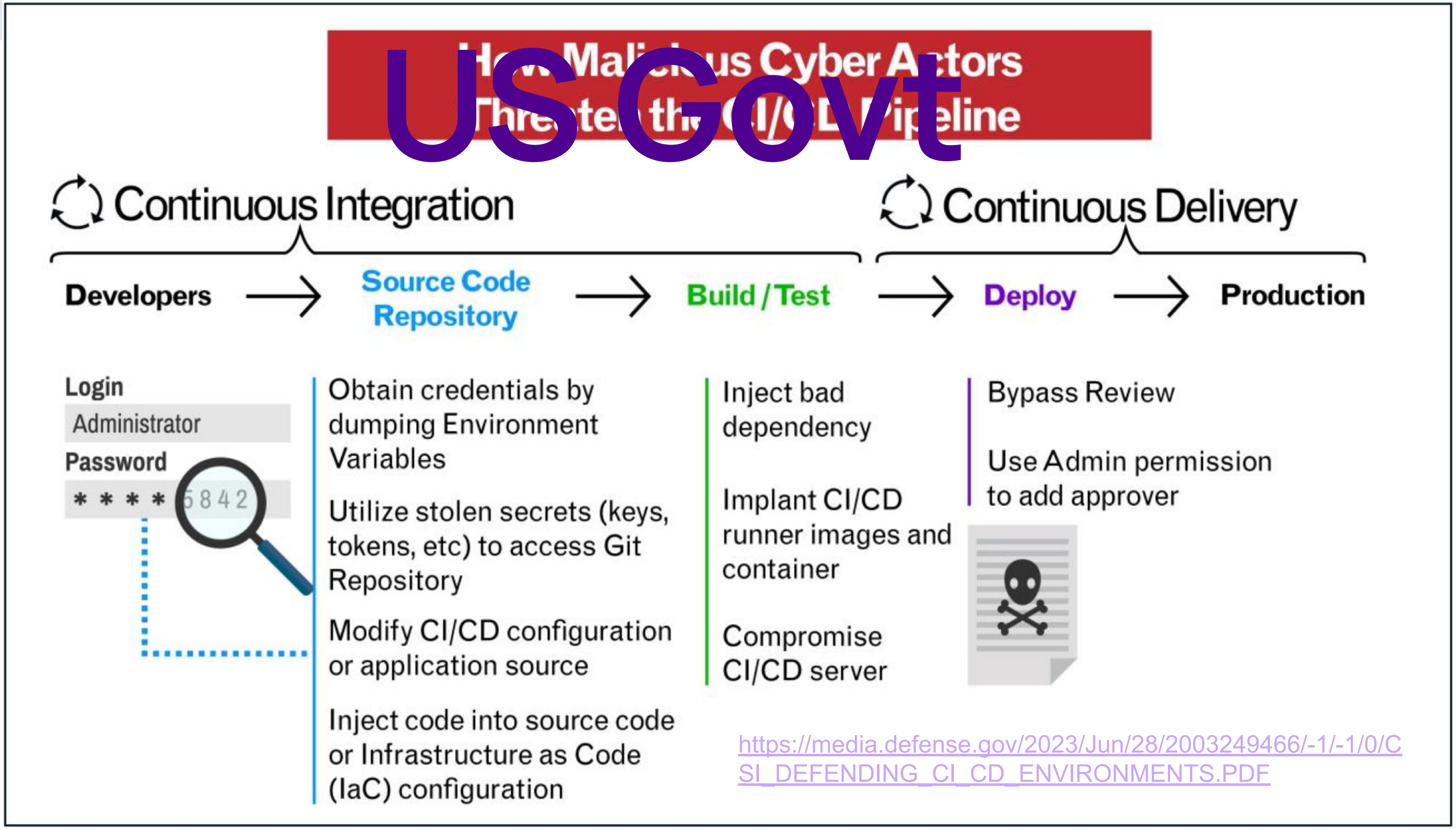
Deploy to Test Envir...
3 jobs completed 7m 18s
1 artifact
1 check passed
do the deploy to test 48s
testAcceptance 4m 25s
dastScanTest 1m 21s
Rerun stage

Deploy to Productio...
4 jobs completed 1h 51m 46s
1 artifact
1 check passed
1 manual validation passed
Wait for external va... 1h 51m...
do the deploy to Produc... 5...
dastScanProd 1m 55s
prodAcceptance 4s
Rerun stage



```yaml
  autoCancel: true

trigger:
  - main

pool:
  vmImage: ubuntu-latest


stages:
  - stage: validate
    displayName: "Ensure that the code works"
    jobs:
      - job: audit
        steps:
          - bash: make audit lint
          - bash: make test
            failOnStderr: false

          Settings
          - task: PublishTestResults@2
            inputs:
              testResultsFormat: JUnit
              testResultsFiles: 'build/reports/report.xml'
```

US Govt



Obtain credentials by dumping Environment Variables

Utilize stolen secrets (keys, tokens, etc) to access Git Repository

Modify CI/CD configuration or application source

Inject code into source code or Infrastructure as Code (IaC) configuration

Inject bad dependency

Implant CI/CD runner images and container

Compromise CI/CD server

Bypass Review

Use Admin permission to add approver

https://media.defense.gov/2023/Jun/28/2003249466/-1/-1/0/CSI_DEFENDING_CI_CD_ENVIRONMENTS.PDF

Figure 1: Threats to the CI/CD pipeline

CI/CD Top 10

safeadvisory.

CI/CD Top 10

- CI/CD has come of age
- Attackers have also matured with the industry
- CI/CD servers are a tempting target for threat actors
- We’ve already seen some serious breaches via CI/CD: Travis, CodeCov, SolarWinds, etc
- Thanks to Daniel Krivelevich and Omer Gil at Cider Security

CICD-SEC-1: Insufficient flow control mechanisms
CICD-SEC-2: Inadequate identity and access management
CICD-SEC-3: Dependency chain abuse
CICD-SEC-4: Poisoned pipeline execution (PPE)
CICD-SEC-5: Insufficient PBAC (pipeline-based access controls)
CICD-SEC-6: Insufficient credential hygiene
CICD-SEC-7: Insecure system configuration
CICD-SEC-8: Ungoverned use of third party services
CICD-SEC-9: Improper artifact integrity validation
CICD-SEC-10: Insufficient logging and visibility

How’s it going to work from here?

risk › control › example

RISK

Insufficient flow control mechanisms

CICD-SEC-1

- Attackers with access to SCM, CI or other systems can deploy malicious artefacts to production without approval or review.
- Code pushes to production, auto-merges of code, malicious artefacts, changes to infrastructure are all possible
- Applicable to Git repos, CI/CD systems, utilities

CONTROLS

Insufficient flow control mechanisms

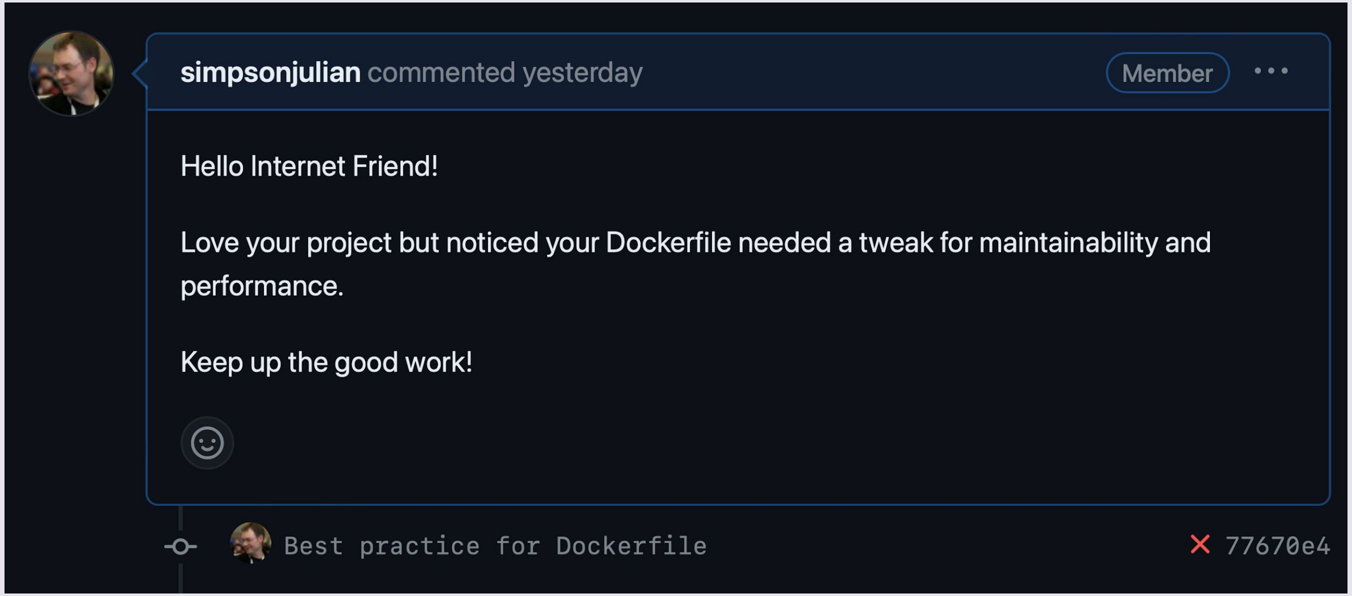
CICD-SEC-1

- Add Branch Protection rules for branches that go to production
- Limit auto-merge rules
- Bake reviews into the pipeline to limit the impact when one gets through
- Use Drift Detection to sniff out config change

EXAMPLE

Insufficient flow control mechanisms

CICD-SEC-1



simpsonjulian commented yesterday

Hello Internet Friend!

Love your project but noticed your Dockerfile needed a tweak for maintainability and performance.

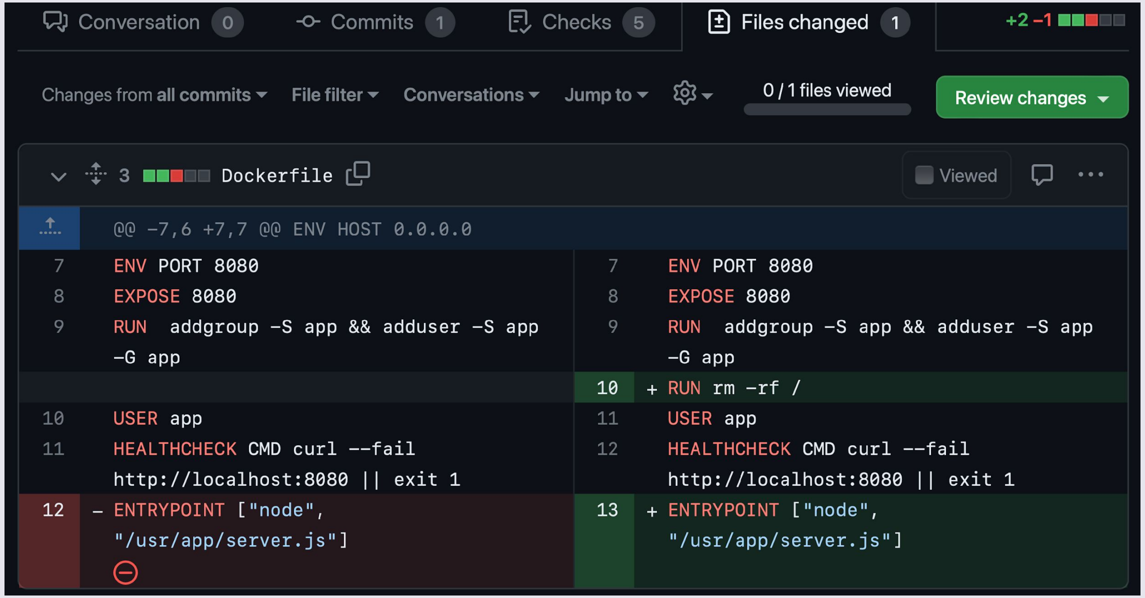
Keep up the good work!

```
Best practice for Dockerfile                                   ✗ 77670e4
```

EXAMPLE

Insufficient flow control mechanisms

CICD-SEC-1



```
@@ -7,6 +7,7 @@ ENV HOST 0.0.0.0
```

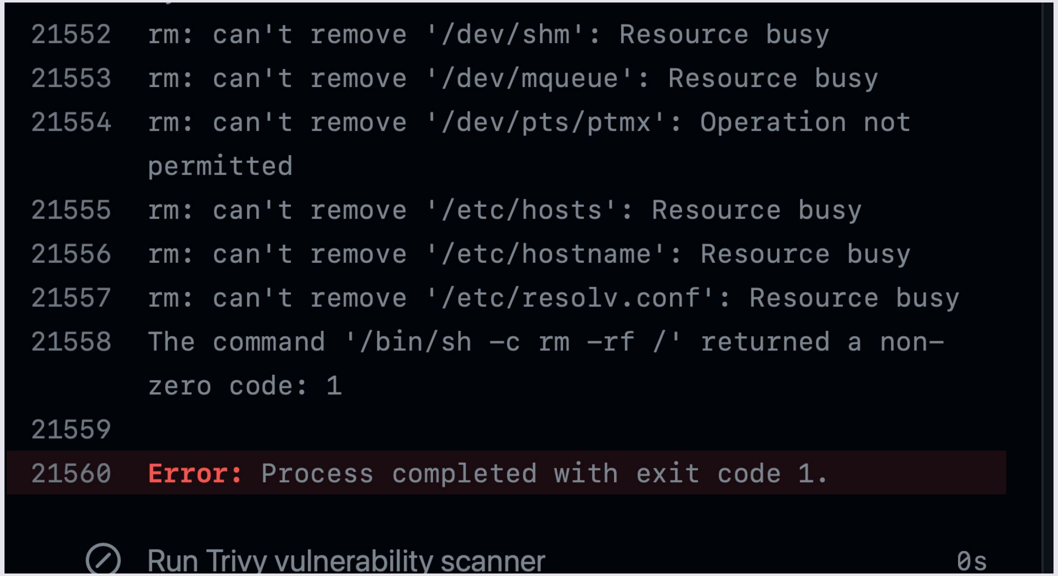
```
ENV PORT 8080                                    ENV PORT 8080
EXPOSE 8080                                      EXPOSE 8080
RUN  addgroup -S app && adduser -S app           RUN  addgroup -S app && adduser -S app
-G app                                           -G app
                                               + RUN rm -rf /
USER app                                         USER app
HEALTHCHECK CMD curl --fail                      HEALTHCHECK CMD curl --fail
http://localhost:8080 || exit 1                  http://localhost:8080 || exit 1
- ENTRYPOINT ["node",                          + ENTRYPOINT ["node",
"/usr/app/server.js"]                            "/usr/app/server.js"]
```

safe advisory.

EXAMPLE

Insufficient flow control mechanisms

CICD-SEC-1

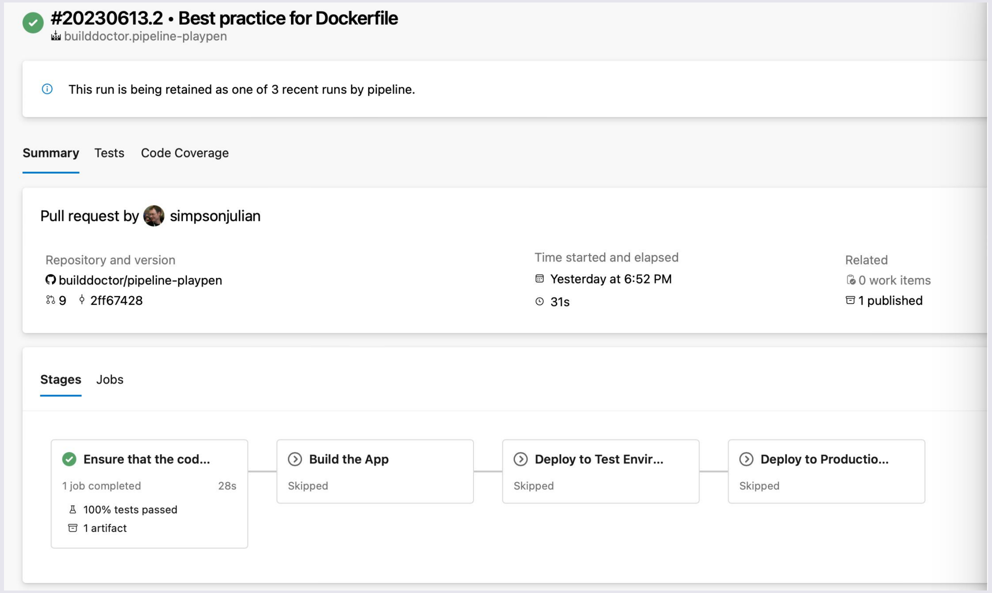


```
rm: can't remove '/dev/shm': Resource busy
rm: can't remove '/dev/mqueue': Resource busy
rm: can't remove '/dev/pts/ptmx': Operation not
permitted
rm: can't remove '/etc/hosts': Resource busy
rm: can't remove '/etc/hostname': Resource busy
rm: can't remove '/etc/resolv.conf': Resource busy
The command '/bin/sh -c rm -rf /' returned a non-
zero code: 1

Error: Process completed with exit code 1.
```

EXAMPLE

Insufficient flow control mechanisms

CICD-SEC-1



#20230613.2 • Best practice for Dockerfile
builddoctor.pipeline-playpen

This run is being retained as one of 3 recent runs by pipeline.

Summary Tests Code Coverage

Pull request by simpsonjulian

Repository and version
builddoctor/pipeline-playpen
9 2ff67428

Time started and elapsed
Yesterday at 6:52 PM
31s

Related
0 work items
1 published

Stages Jobs

Ensure that the cod...
1 job completed 28s
100% tests passed
1 artifact

Build the App
Skipped

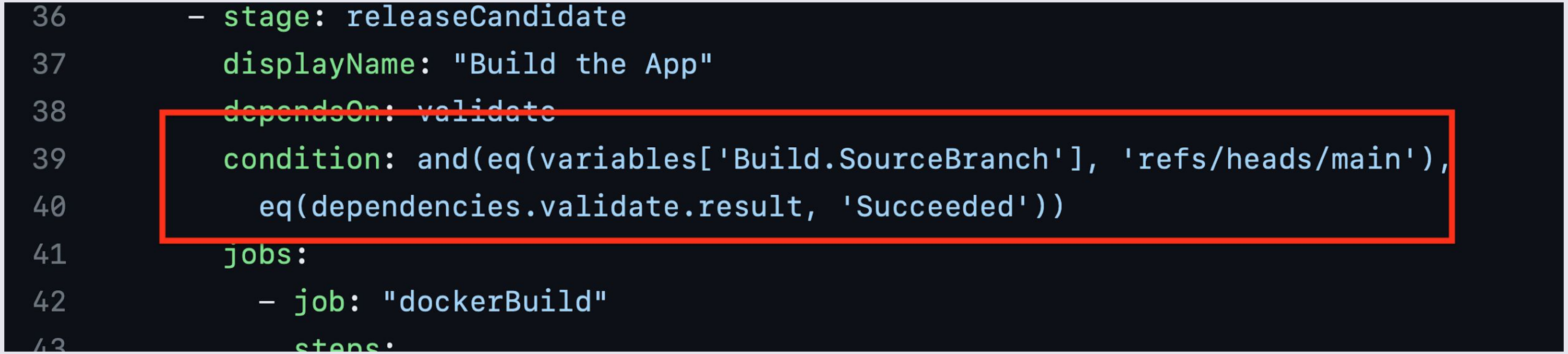
Deploy to Test Envir...
Skipped

Deploy to Productio...
Skipped

EXAMPLE

Insufficient flow control mechanisms

CICD-SEC-1



```
- stage: releaseCandidate
  displayName: "Build the App"
  dependsOn: validate
  condition: and(eq(variables['Build.SourceBranch'], 'refs/heads/main'),
    eq(dependencies.validate.result, 'Succeeded'))
  jobs:
    - job: "dockerBuild"
      steps:
```

EXAMPLE

Insufficient flow control mechanisms

CICD-SEC-1

```
displayName: "Deploy to Production Environment"
dependsOn: testEnv
jobs:
  - job: waitForValidation
    displayName: Wait for external validation
    pool: server
    timeoutInMinutes: 4320
    steps:
      - task: ManualValidation@0
        timeoutInMinutes: 1440
        inputs:
          notifyUsers: simpsonjulian@gmail.com
          instructions: 'Please validate the build configuration and resume'
          onTimeout: 'reject'
```

RISK

Dependency chain abuse

CICD-SEC-3

- Dependency confusion (public packages that attempt to mimic your org's private packages)
- Dependency hijacking (taking control of a public package that your org uses)
- Typosquatting (seeding package repos with common misspellings of popular packages)
- Brandjacking (borrowing credibility from a brand so that developers trust your packages)
- **New!** NPM Manifest Confusion - *package.json* in GitHub is not the *package.json* in the distributed package

CONTROLS

Dependency chain abuse

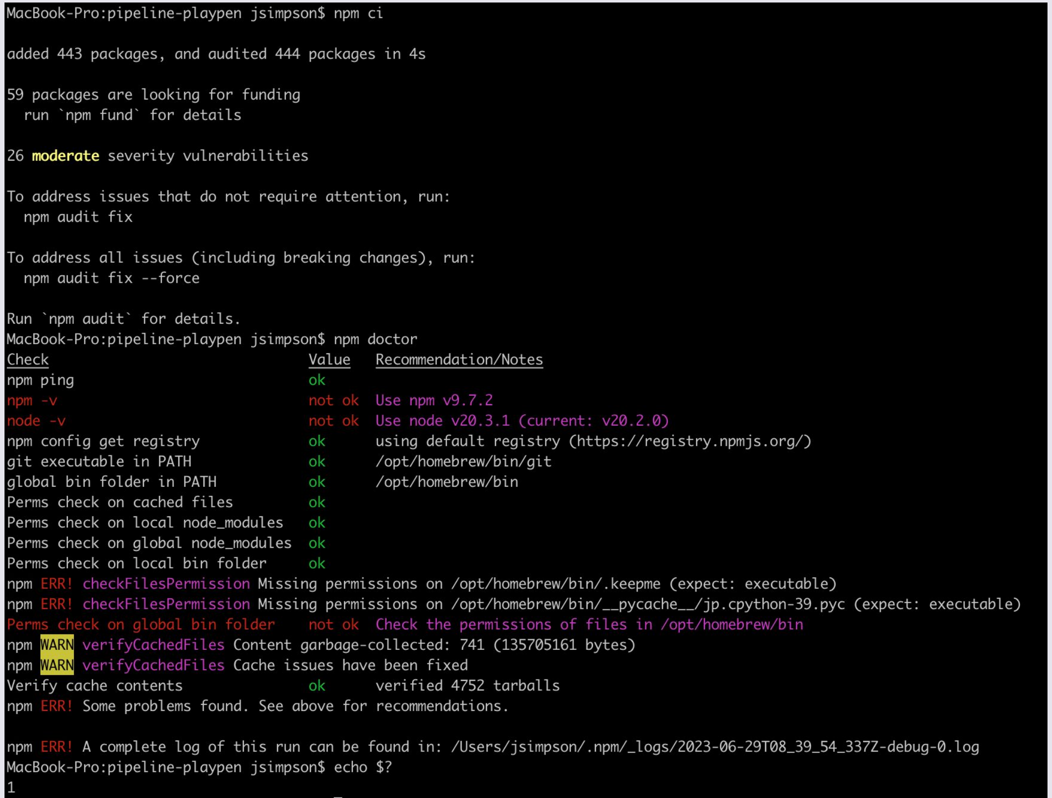
CICD-SEC-3

- Proxy all packages from the internet, instead of having dev systems fetch direct
- Encourage the use of internal, pre-approved packages where possible
- Enable checksums and signatures
- Register and document package scopes for your org to reduce confusion

EXAMPLE

Dependency chain abuse

CICD-SEC-3

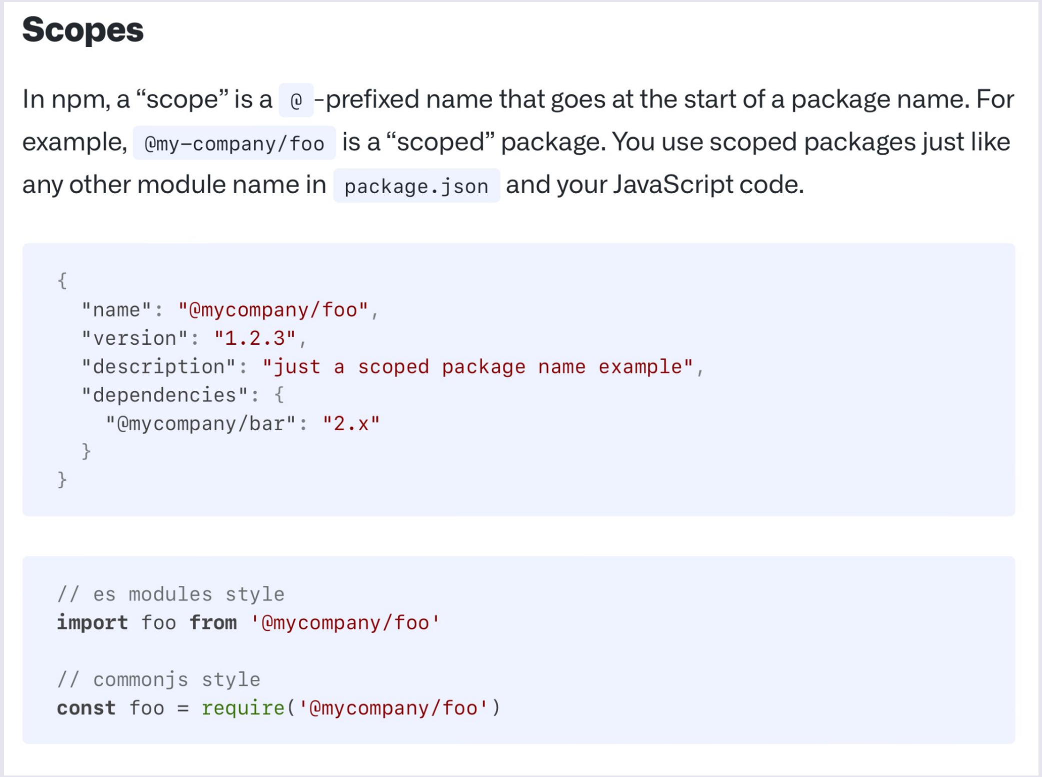


```
MacBook-Pro:pipeline-playpen jsimpson$ npm ci

added 443 packages, and audited 444 packages in 4s

59 packages are looking for funding
  run `npm fund` for details

26 moderate severity vulnerabilities

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
MacBook-Pro:pipeline-playpen jsimpson$ npm doctor
Check                               Value   Recommendation/Notes
npm ping                            ok
npm -v                              not ok  Use npm v9.7.2
node -v                             not ok  Use node v20.3.1 (current: v20.2.0)
npm config get registry             ok      using default registry (https://registry.npmjs.org/)
git executable in PATH              ok      /opt/homebrew/bin/git
global bin folder in PATH           ok      /opt/homebrew/bin
Perms check on cached files         ok
Perms check on local node_modules   ok
Perms check on global node_modules  ok
Perms check on local bin folder     ok
npm ERR! checkFilesPermission Missing permissions on /opt/homebrew/bin/.keepme (expect: executable)
npm ERR! checkFilesPermission Missing permissions on /opt/homebrew/bin/__pycache__/jp.cpython-39.pyc (expect: executable)
Perms check on global bin folder    not ok  Check the permissions of files in /opt/homebrew/bin
npm WARN verifyCachedFiles Content garbage-collected: 741 (135705161 bytes)
npm WARN verifyCachedFiles Cache issues have been fixed
Verify cache contents               ok      verified 4752 tarballs
npm ERR! Some problems found. See above for recommendations.

npm ERR! A complete log of this run can be found in: /Users/jsimpson/.npm/_logs/2023-06-29T08_39_54_337Z-debug-0.log
MacBook-Pro:pipeline-playpen jsimpson$ echo $?
1
```

EXAMPLE

Dependency chain abuse

CICD-SEC-3



Scopes

In npm, a “scope” is a `@` -prefixed name that goes at the start of a package name. For example, `@my-company/foo` is a “scoped” package. You use scoped packages just like any other module name in `package.json` and your JavaScript code.

```
{
  "name": "@mycompany/foo",
  "version": "1.2.3",
  "description": "just a scoped package name example",
  "dependencies": {
    "@mycompany/bar": "2.x"
  }
}
```

```
// es modules style
import foo from '@mycompany/foo'

// commonjs style
const foo = require('@mycompany/foo')
```

safe advisory.

RISK

Poisoned pipeline execution (PPE)

CICD-SEC-4

Attackers with access to code (and not CI/CD system) can still manipulate the build process:

- **Direct PPE:** attacker modifies config files in repository, via direct commit to unprotected branch, or via Pull Request
- **Indirect PPE:** attacker injects malicious code via build systems, build scripts, test frameworks or automatic tools like linters and scanners
- **Public PPE:** many projects build in public and see Pull Requests as a necessary part of Open Source development. Malicious OSS participation can expose secrets or other code.

CONTROLS

Poisoned pipeline execution (PPE)

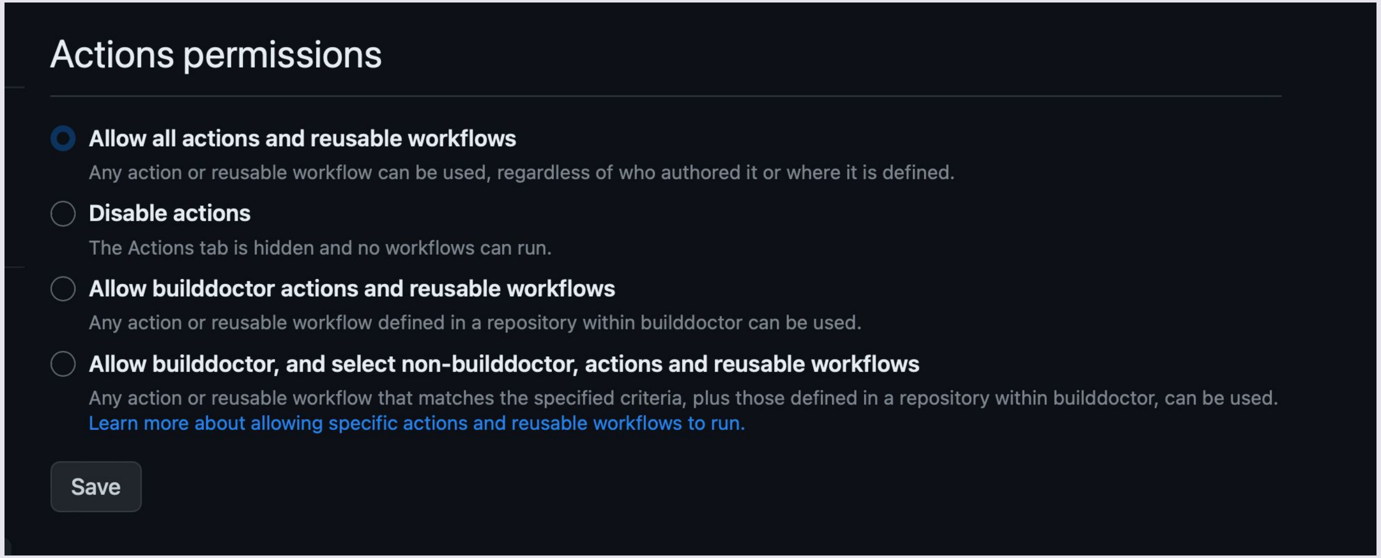
CICD-SEC-4

- Ensure that pipelines exposed to unreviewed code run on isolated nodes, not the ones that connect to everything else (e.g. some else's)
- If you must build from public repos: where possible, don't build from forks
- Ensure there are branch protection rules to limit triggers
- Try and keep pipeline configuration away from exposed repos
- Limit SCM access to those who really need it
- Limit credentials granted to pipelines

EXAMPLE

Poisoned pipeline execution (PPE)

CICD-SEC-4



Actions permissions

[✓ **Allow all actions and reusable workflows**]
Any action or reusable workflow can be used, regardless of who authored it or where it is defined.

Disable actions
The Actions tab is hidden and no workflows can run.

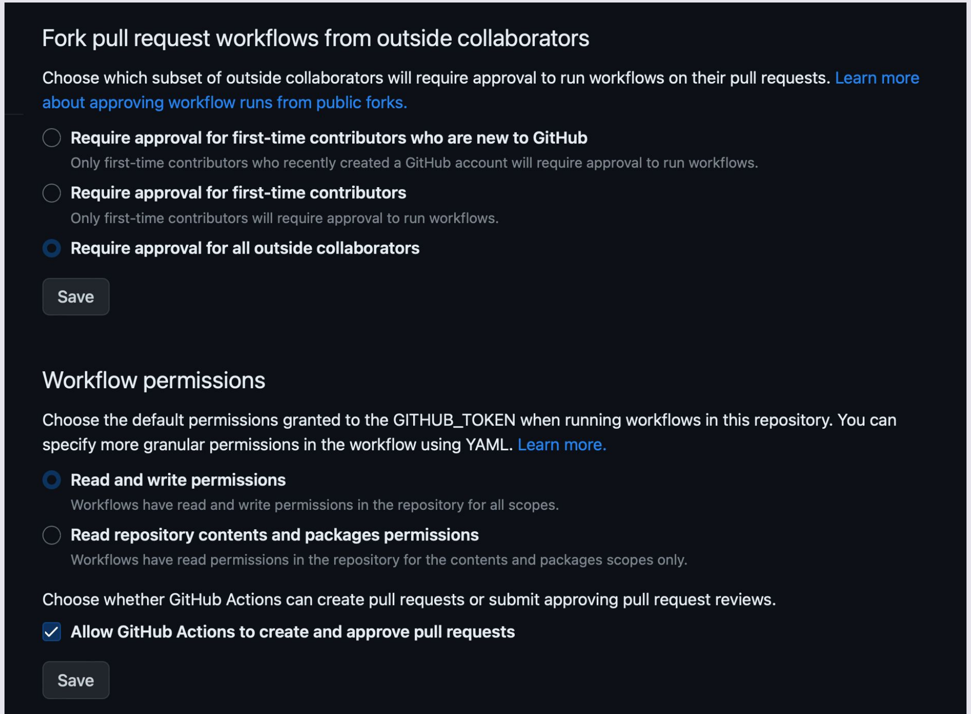
Allow builddoctor actions and reusable workflows
Any action or reusable workflow defined in a repository within builddoctor can be used.

Allow builddoctor, and select non-builddoctor, actions and reusable workflows
Any action or reusable workflow that matches the specified criteria, plus those defined in a repository within builddoctor, can be used.
Learn more about allowing specific actions and reusable workflows to run.

EXAMPLE

Poisoned pipeline execution (PPE)

CICD-SEC-4



Fork pull request workflows from outside collaborators

Choose which subset of outside collaborators will require approval to run workflows on their pull requests. Learn more about approving workflow runs from public forks.

- [ ] **Require approval for first-time contributors who are new to GitHub**
  Only first-time contributors who recently created a GitHub account will require approval to run workflows.
- [ ] **Require approval for first-time contributors**
  Only first-time contributors will require approval to run workflows.
- [✓] **Require approval for all outside collaborators**

Workflow permissions

Choose the default permissions granted to the GITHUB_TOKEN when running workflows in this repository. You can specify more granular permissions in the workflow using YAML. Learn more.

- [✓] **Read and write permissions**
  Workflows have read and write permissions in the repository for all scopes.
- [ ] **Read repository contents and packages permissions**
  Workflows have read permissions in the repository for the contents and packages scopes only.

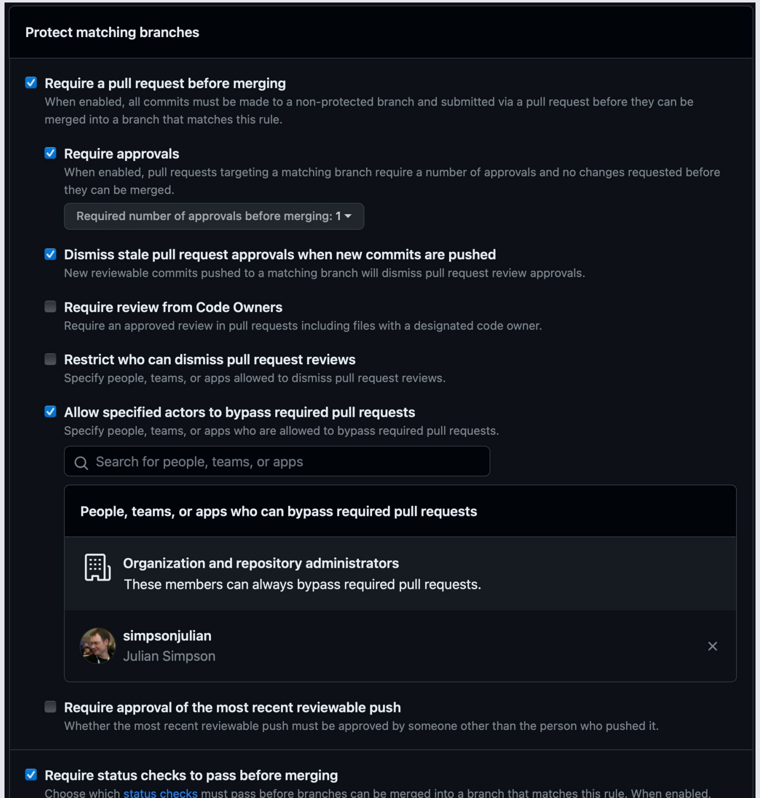
Choose whether GitHub Actions can create pull requests or submit approving pull request reviews.

- [✓] **Allow GitHub Actions to create and approve pull requests**

EXAMPLE

Poisoned pipeline execution (PPE)

CICD-SEC-4



Protect matching branches

☑ **Require a pull request before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☑ **Require approvals**
When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.

Required number of approvals before merging: 1

☑ **Dismiss stale pull request approvals when new commits are pushed**
New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

☐ **Require review from Code Owners**
Require an approved review in pull requests including files with a designated code owner.

☐ **Restrict who can dismiss pull request reviews**
Specify people, teams, or apps allowed to dismiss pull request reviews.

☑ **Allow specified actors to bypass required pull requests**
Specify people, teams, or apps who are allowed to bypass required pull requests.

Search for people, teams, or apps

People, teams, or apps who can bypass required pull requests

Organization and repository administrators
These members can always bypass required pull requests.

simpsonjulian
Julian Simpson

☐ **Require approval of the most recent reviewable push**
Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

☑ **Require status checks to pass before merging**
Choose which status checks must pass before branches can be merged into a branch that matches this rule. When enabled,

safe advisory.

RISK

Insufficient PBAC (Pipeline-Based Access Controls)

CICD-SEC-5

- Pipelines need to run on someone's computer (even in the cloud)
- That node will access source code, cloud services, secrets, artifacts, filesystems, other pipelines, SSH keys, the public Internet, your network
- Someone needs to have a good think about what could go wrong

CONTROL

Insufficient PBAC (Pipeline-Based Access Controls)

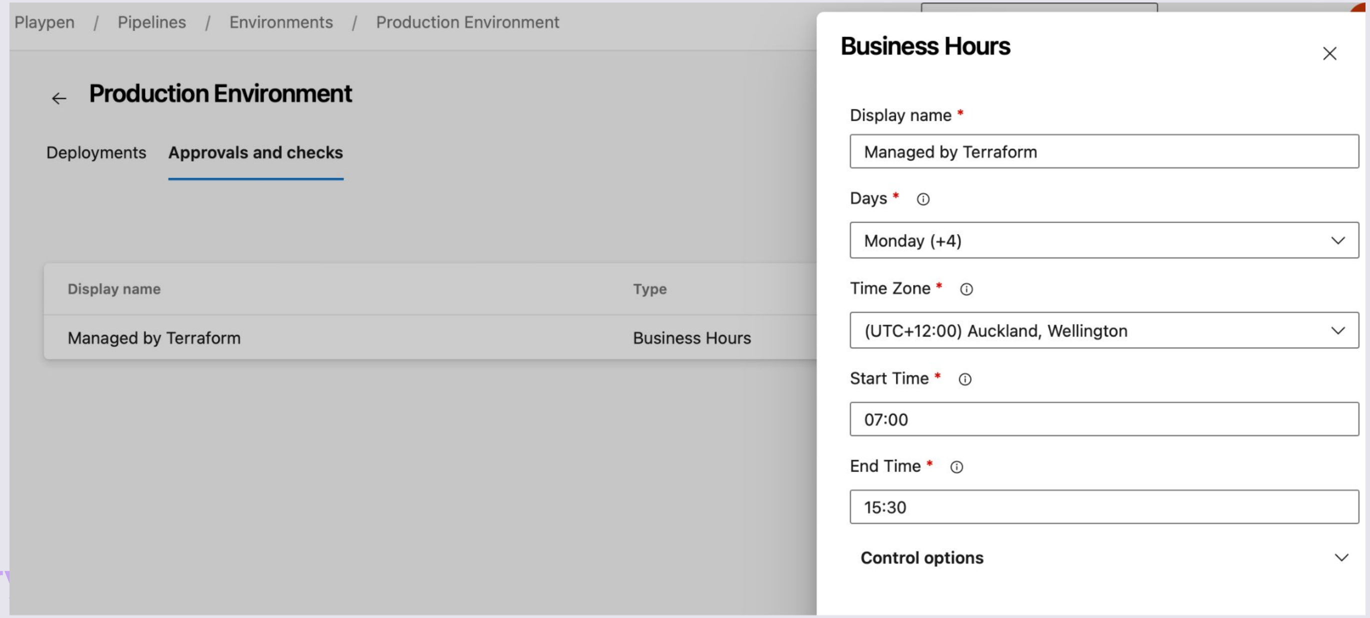
CICD-SEC-5

- Each pipeline should have enough access to resources needed to do its job and no more
- That includes nodes, which should also have access controls to prevent lateral movement and data breaches (does your pipeline have full access to the production database?)
- ADO users, this is where you should put Checks on Service Connections, Pipelines, Environments

CONTROL

Insufficient PBAC (Pipeline-Based Access Controls)

CICD-SEC-5



Playpen / Pipelines / Environments / Production Environment

Production Environment

Deployments | Approvals and checks

Display name	Type
Managed by Terraform	Business Hours

Business Hours

Display name *: Managed by Terraform

Days *: Monday (+4)

Time Zone *: (UTC+12:00) Auckland, Wellington

Start Time *: 07:00

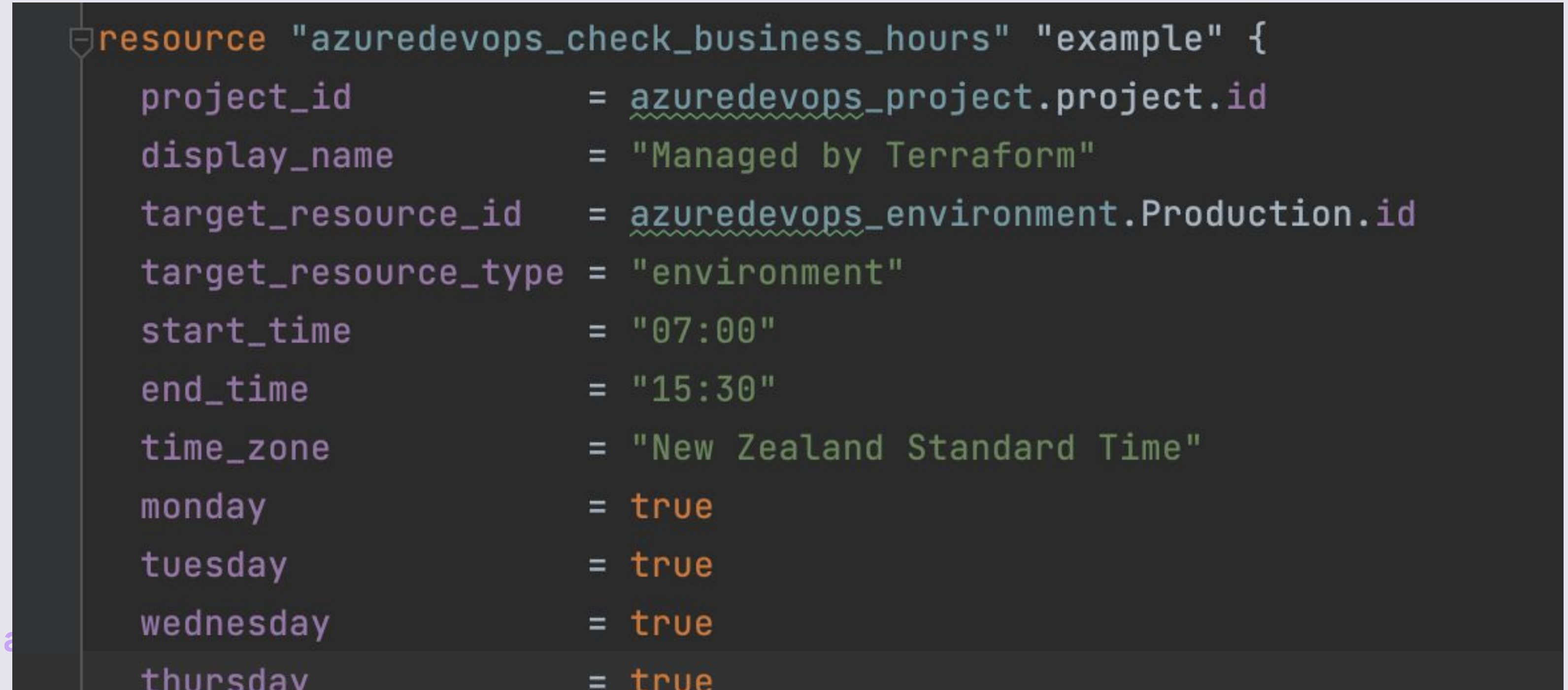
End Time *: 15:30

Control options

safe advisor

CONTROL

Insufficient PBAC (Pipeline-Based Access Controls)



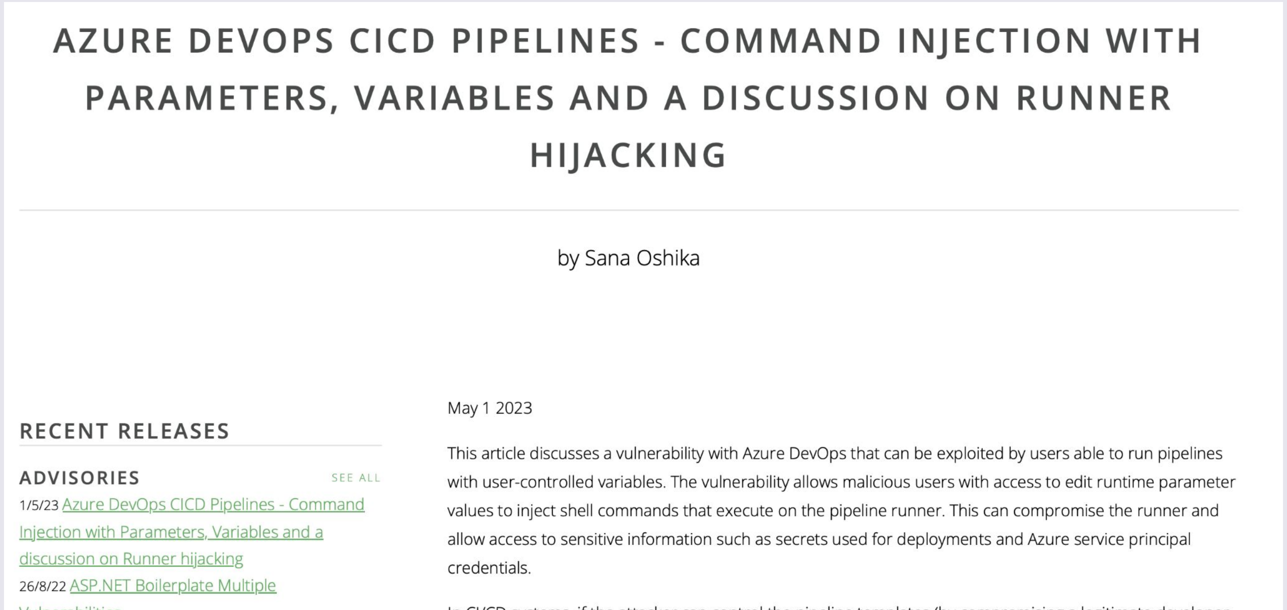
```
resource "azuredevops_check_business_hours" "example" {
  project_id           = azuredevops_project.project.id
  display_name         = "Managed by Terraform"
  target_resource_id   = azuredevops_environment.Production.id
  target_resource_type = "environment"
  start_time           = "07:00"
  end_time             = "15:30"
  time_zone            = "New Zealand Standard Time"
  monday               = true
  tuesday              = true
  wednesday            = true
  thursday             = true
```

safe

EXAMPLE

Insufficient PBAC (Pipeline-Based Access Controls)

CICD-SEC-5



AZURE DEVOPS CICD PIPELINES - COMMAND INJECTION WITH PARAMETERS, VARIABLES AND A DISCUSSION ON RUNNER HIJACKING

by Sana Oshika

RECENT RELEASES

ADVISORIES SEE ALL

1/5/23 Azure DevOps CICD Pipelines - Command Injection with Parameters, Variables and a discussion on Runner hijacking

26/8/22 ASP.NET Boilerplate Multiple

May 1 2023

This article discusses a vulnerability with Azure DevOps that can be exploited by users able to run pipelines with user-controlled variables. The vulnerability allows malicious users with access to edit runtime parameter values to inject shell commands that execute on the pipeline runner. This can compromise the runner and allow access to sensitive information such as secrets used for deployments and Azure service principal credentials.

safe advisory.

RISK

Improper artifact integrity validation

CICD-SEC-9

- Otherwise known by its celebrity alter-ego, the SolarWinds attack.
- An adversary can alter or inject artefacts with a malicious payload

CONTROL

Improper artifact integrity validation

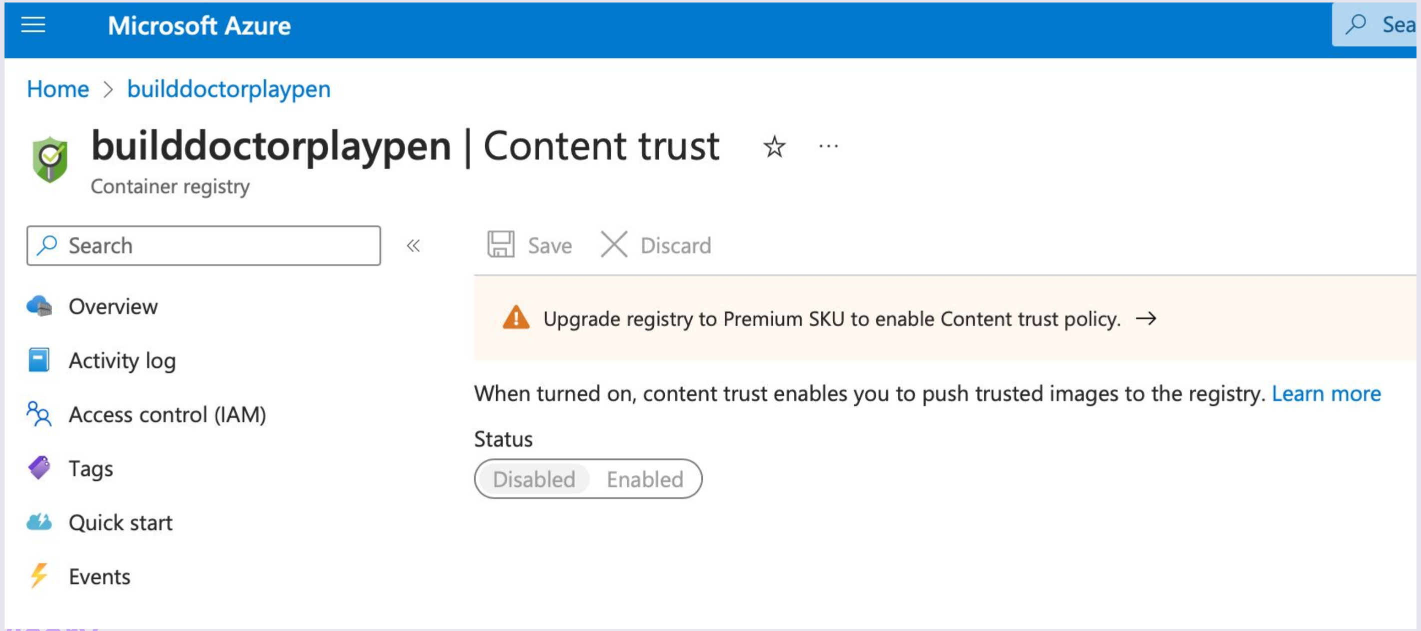
CICD-SEC-9

- Consider commit signing
- Verification tools for artefacts, e.g. signing
- Manage configuration drift
- Third party resources should always be validated - review the full dependency chain for your pipelines

EXAMPLE

Improper artifact integrity validation

CICD-SEC-9



builddoctorplaypen | Content trust

Container registry

Upgrade registry to Premium SKU to enable Content trust policy. →

When turned on, content trust enables you to push trusted images to the registry. Learn more

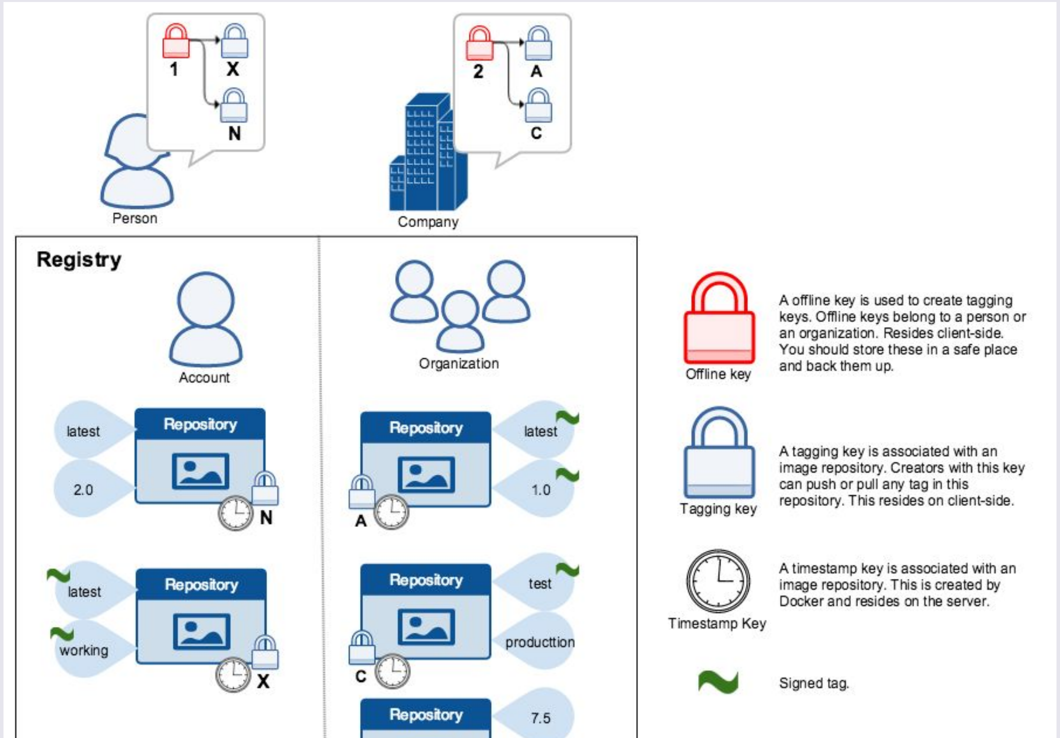
Status

[✓ Disabled] Enabled

EXAMPLE

Improper artifact integrity validation

CICD-SEC-9


1
X
N
Person
2
A
C
Company
Registry
Account
Organization
latest
2.0
Repository
N
Repository
latest
1.0
A
latest
working
Repository
X
Repository
test
producttion
C
Repository
7.5
Offline key
A offline key is used to create tagging keys. Offline keys belong to a person or an organization. Resides client-side. You should store these in a safe place and back them up.
Tagging key
A tagging key is associated with an image repository. Creators with this key can push or pull any tag in this repository. This resides on client-side.
Timestamp Key
A timestamp key is associated with an image repository. This is created by Docker and resides on the server.
Signed tag.

and all the rest...

(the quickfire round)

Inadequate IAM

CICD-SEC-2

- It's necessary to connect the CI/CD tooling to just about everything
- Identities are sprinkled throughout the CI/CD ecosystem: Git, CI/CD, container registries, app servers, APM tools, etc.
- Least privilege often goes out the window with delivery deadlines
- Stale, local, external, self-registered or shared identities make it worse

Insufficient credential hygiene

CICD-SEC-6

- Credentials are everywhere in a CI/CD system
- They should never be in source (but of course they are)
- They should also be granted for a particular context and not reused
- Nor should they exist in container image layers, or console output
- Credentials should be rotated or retired appropriately

Insecure system configuration

CICD-SEC-7

- CI/CD systems need hardening like (if not more than) production systems
- Patching
- Network access control (what databases can they see?)
- Granting least privilege on the host OS
- Configuration for authorisation, access control, logging etc.
- Credential hygiene

Ungoverned use of third party services

CICD-SEC-8

- Your colleagues can sign up and implement services in minutes
- For example: code analysis tools, testing tools, deployment tools etc
- Third party tools have been compromised and used for attacks on their users

Insufficient logging and visibility

CICD-SEC-10

These systems tend to start with “it’s just a dev tool bro - you’re probably missing the detective controls that you need

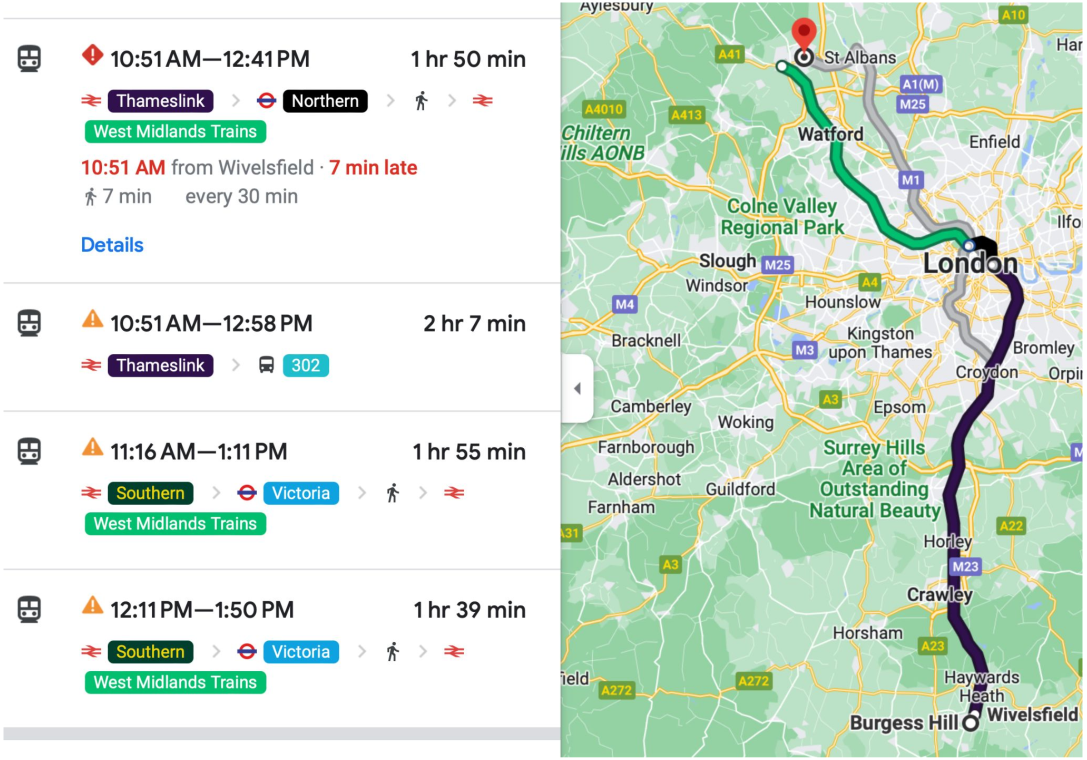
If your threat model doesn’t include ‘what if someone compromises our pipelines or CI/CD infrastructure’, you may not have sufficient:

- Audit logs
- Metrics
- Anomaly detection and SIEM

TL;DR

https://owasp.org/www-project-top-10-ci-cd-security-risks/

Roundabouts



10:51 AM—12:41 PM 1 hr 50 min

Thameslink > Northern > > West Midlands Trains

10:51 AM from Wivelsfield · 7 min late

7 min every 30 min

Details

10:51 AM—12:58 PM 2 hr 7 min

Thameslink > 302

11:16 AM—1:11 PM 1 hr 55 min

Southern > Victoria > > West Midlands Trains

12:11 PM—1:50 PM 1 hr 39 min

Southern > Victoria > > West Midlands Trains

Map data ©2023 Google

safe advisory.

Roundabouts

Obukit, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa34.0>, via Wikipedia

A final plea

- Keep all your pipelines as YAML versioned in project repos
- Keep your YAML valid with a formatter
- Don’t split build and release concerns, unless you must
- Maintain a threat model for your pipelines, nodes and services
- Teach developers about the threats - they have different incentives, but they won’t like a breach either

Thank you!