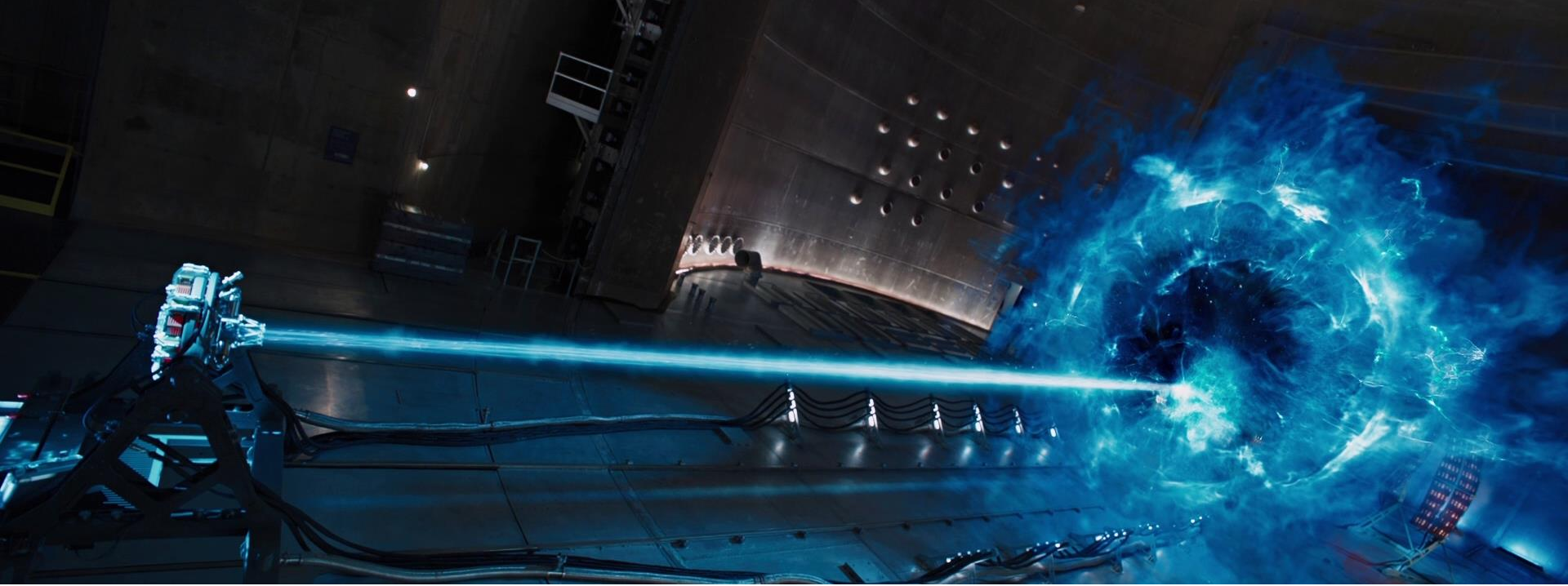


From Zero to Secure Continuous Deployment in 60 Minutes

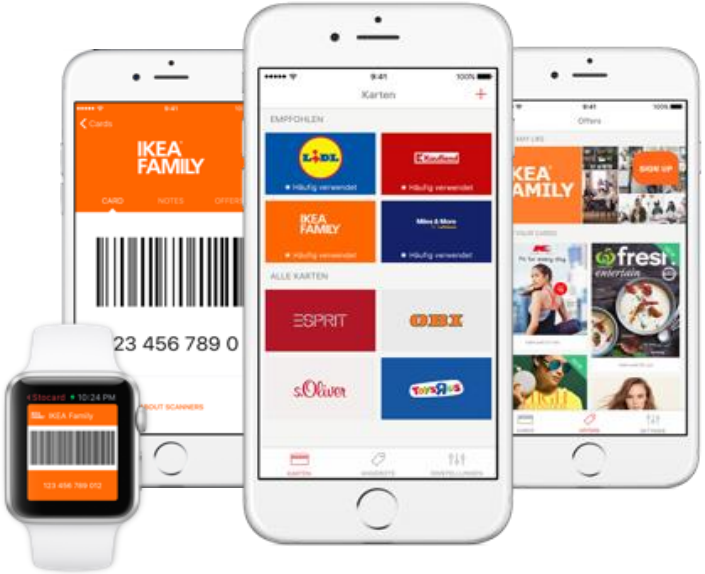
Florian Barth

Matthias Luft

STOCARD

- THE Digital Wallet
- 20.000.000 users
  50 countries
  5 offices

- 25.000.000,000 rpm (onth ;))
  100 micro services
  10 servers
  5 persons doing backend DevOps



- Vendor-independent
- Established 2001
- 65 employees, 42 FTE consultants
- Continuous growth in revenue/profits
  - No venture/equity capital, no external financial obligations of any kind
- Customers predominantly large/very large enterprises
  - Industry, telecommunications, finance

Avengers, assemble!

Captain Security

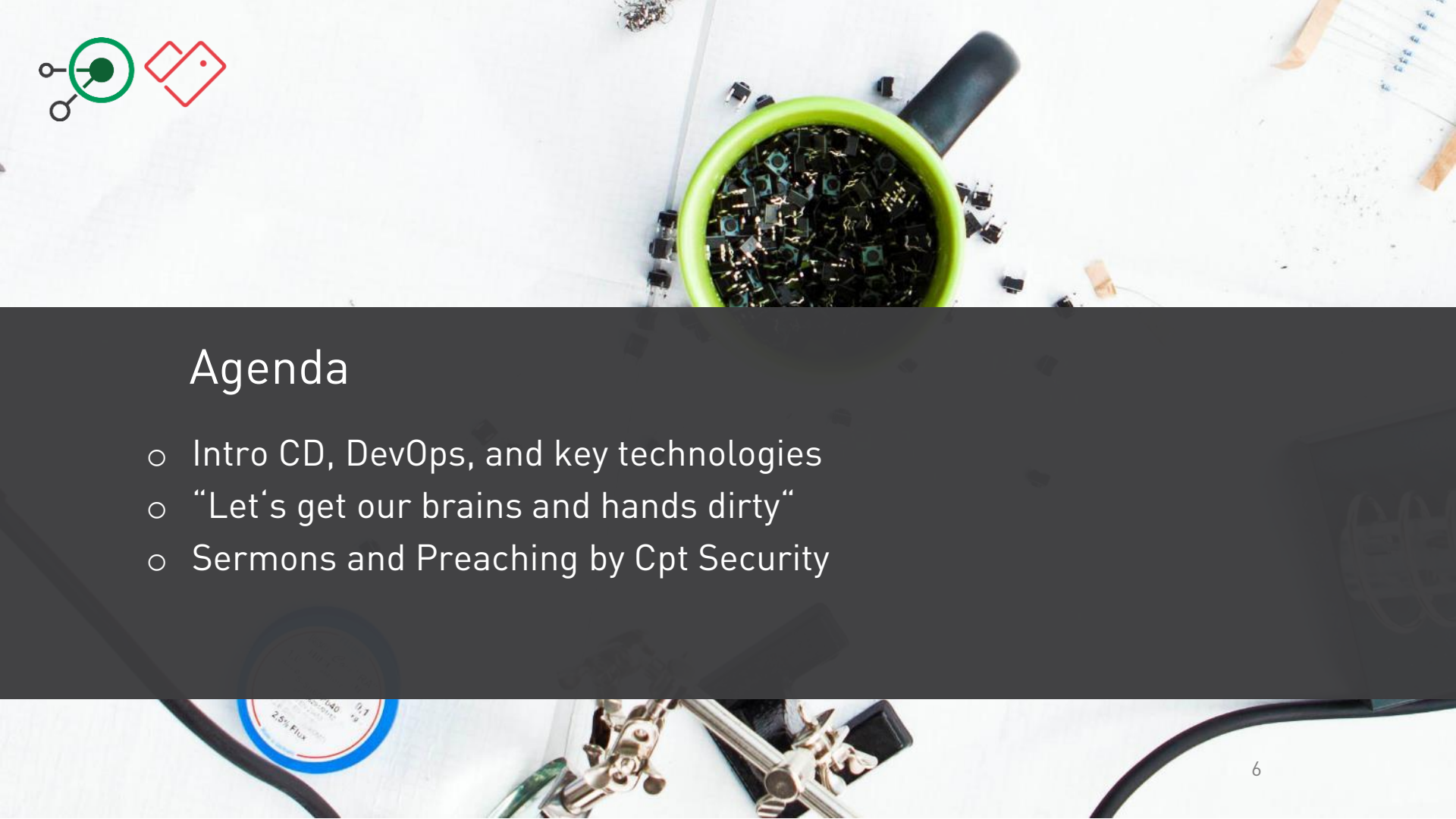
DareDeveloper

I dare you to create an IaC-based automated CD pipeline!

Give me 60 minutes!

You have 30, as I will need to secure that thing!

I'll do it in 10!



Agenda

- Intro CD, DevOps, and key technologies
- “Let‘s get our brains and hands dirty“
- Sermons and Preaching by Cpt Security

DevOps

“DevOps is the philosophy of unifying Development and Operations at the **culture**, **system**, **practice**, and **tool** levels, to achieve **accelerated** and more **frequent delivery** of **value** to the customer, by improving quality in order to increase velocity.”

Rob England, 2014

Continuous Delivery

“**Continuous Delivery (CD)** is a software engineering approach in which teams produce software in **short cycles**, ensuring that the software can be **reliably released** at any time. It aims at **building**, **testing**, and **releasing** software **faster and more frequently**.”

DOI: 10.1109/MS.2015.27

Continuous Deployment

… is often confused with Continuous Delivery and “means that every change goes through the pipeline and **automatically** gets put into production, resulting in **many production deployments every day**.”

https://martinfowler.com/bliki/ContinuousDelivery.html

Immutable Infrastructure

- “Servers are cattle not pets”

Immutable Infrastructure

- ~~"Servers are cattle not pets"~~
- "Servers are **syringes**"

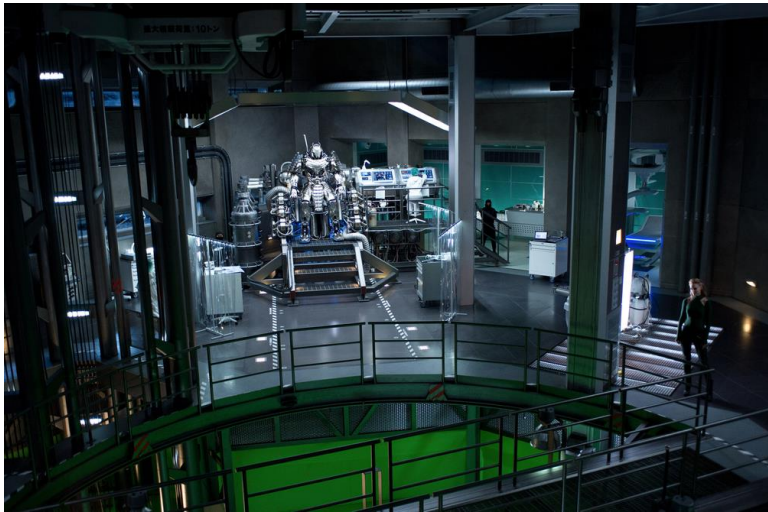
- Spawn your infra from a template or recipe

- **Update**, **Test** it, **Spawn** new, **Trash** old

Infrastructure-as-Code

- Manage und provision **data centers** through **machine-readable** definition files
- Version control your infrastructure

- Documentation & Evolution
- Static/Dynamic analysis



Prepare to be steamrolled…

- A lot of content and prerequisites
- We want to explain the actually difficult parts, not the parts that are routine work.

Goal

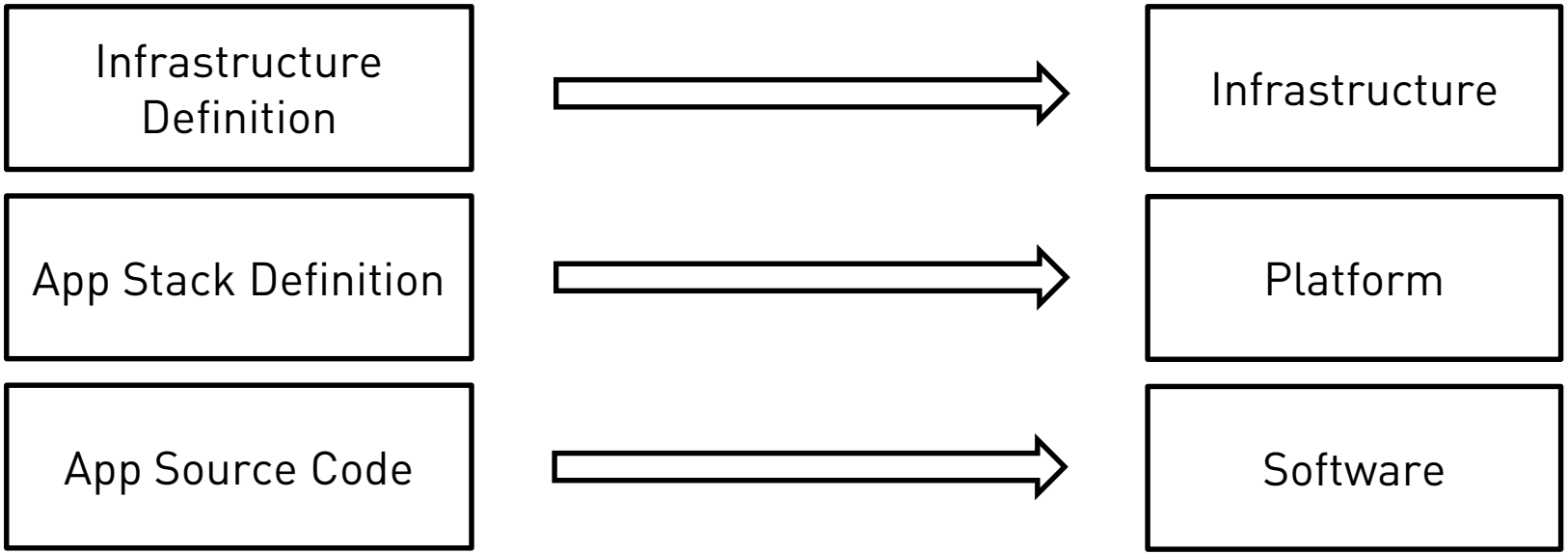
- See the buzzwords in action
- Gain a general understanding

My First Contact with DevOps

- Role: Unifying Dev, Ops, Net, Sec, …
- Scale: users growing exponentially
- Infra: 5€ “vServer”
- Stack/Tooling: nginx, php, ssh, vi
- CD: “**:w**”

- Since then we learned a lot!

Ingredients

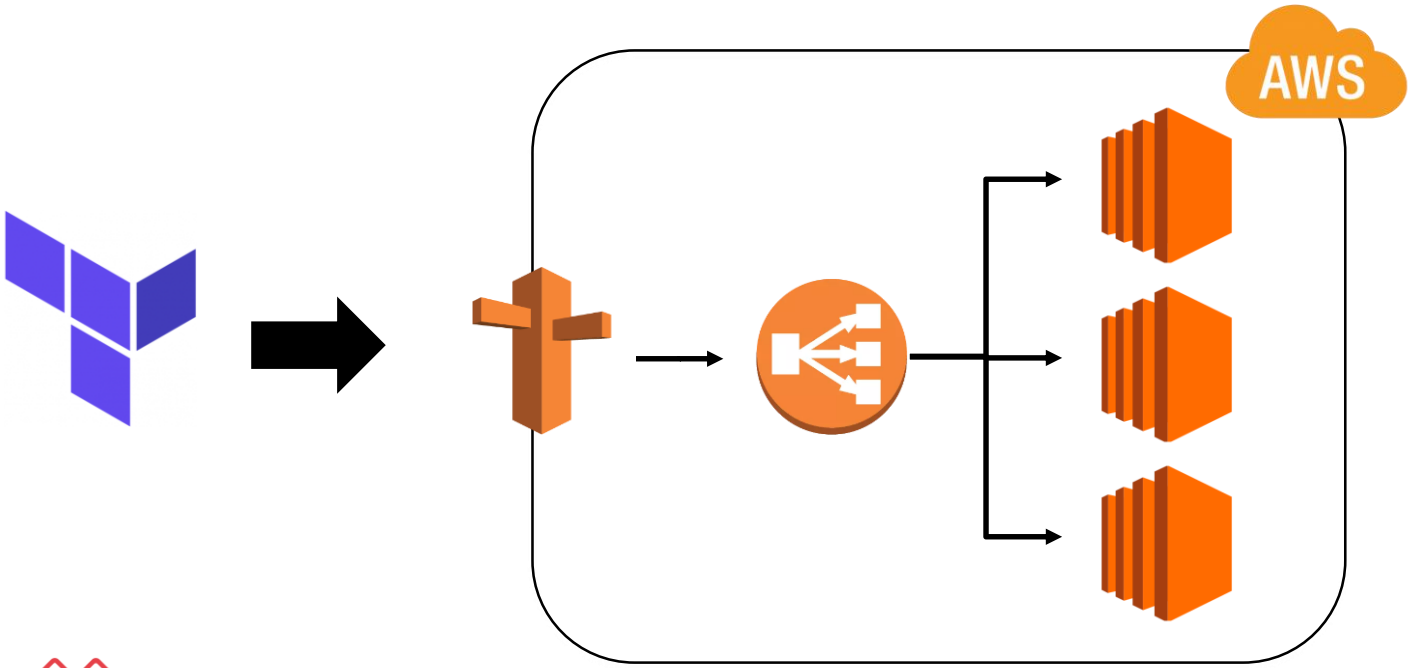


WARNING

The Following stunts are performed by idiots or under the supervision of stupider idiots. If you attempt to reenact any of the idiotic stunts you will see, you will be left with idiotic scars.

Enjoy.

This is DareDeveloper, Welcome to Infrastructure as Code


AWS

Demo

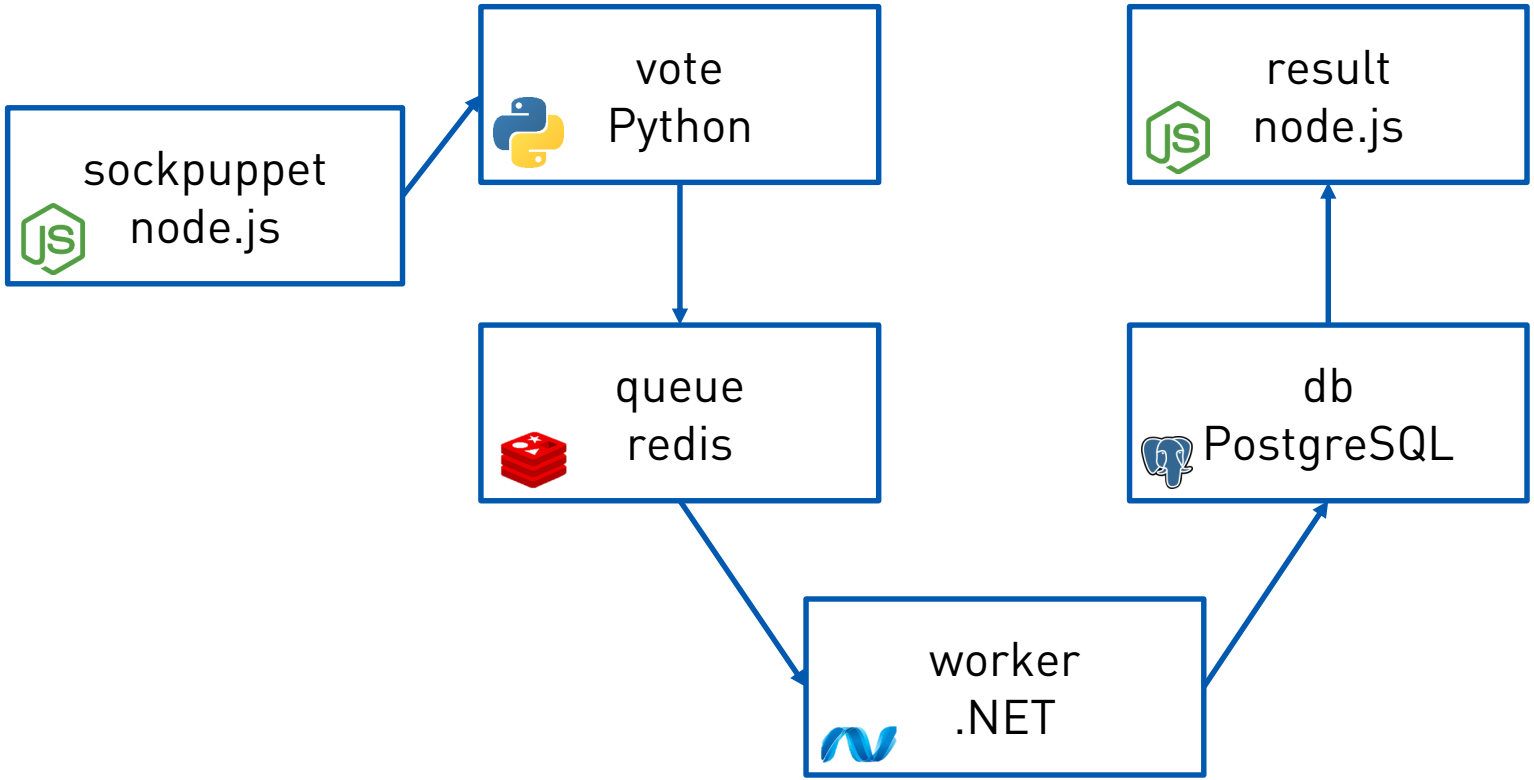
Terraform

WARNING

The Following stunts are performed by idiots or under the supervision of stupider idiots. If you attempt to reenact any of the idiotic stunts you will see, you will be left with idiotic scars.

Enjoy.

This is DareDeveloper, Welcome to App Stack as Code


sockpuppet
node.js
vote
Python
result
node.js
queue
redis
db
PostgreSQL
worker
.NET

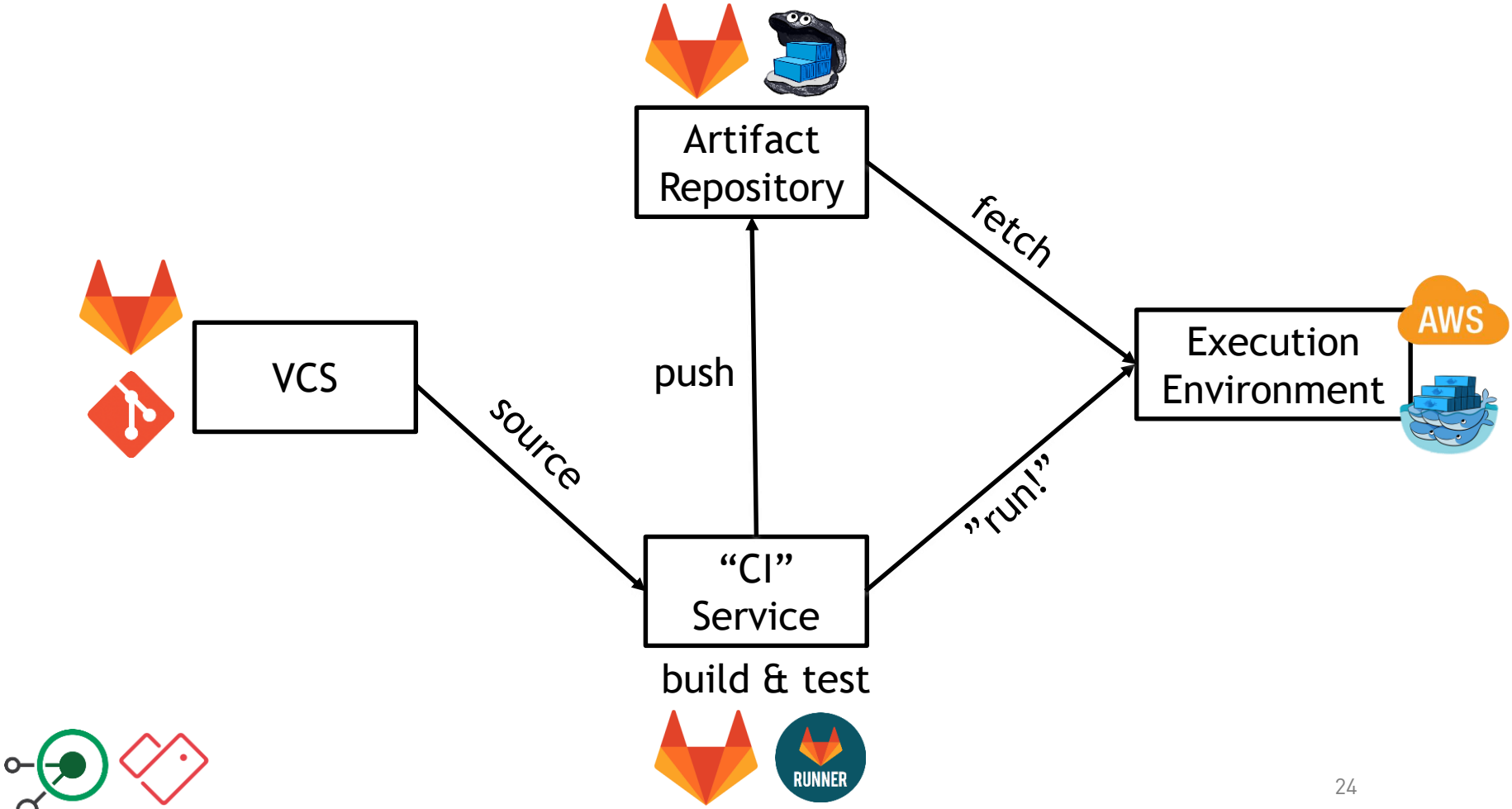
Demo

Docker Compose
Service Infra

DevOps Borat
@DEVOPS_BORAT

To make error is human. To propagate error to all server in automatic way is #devops.

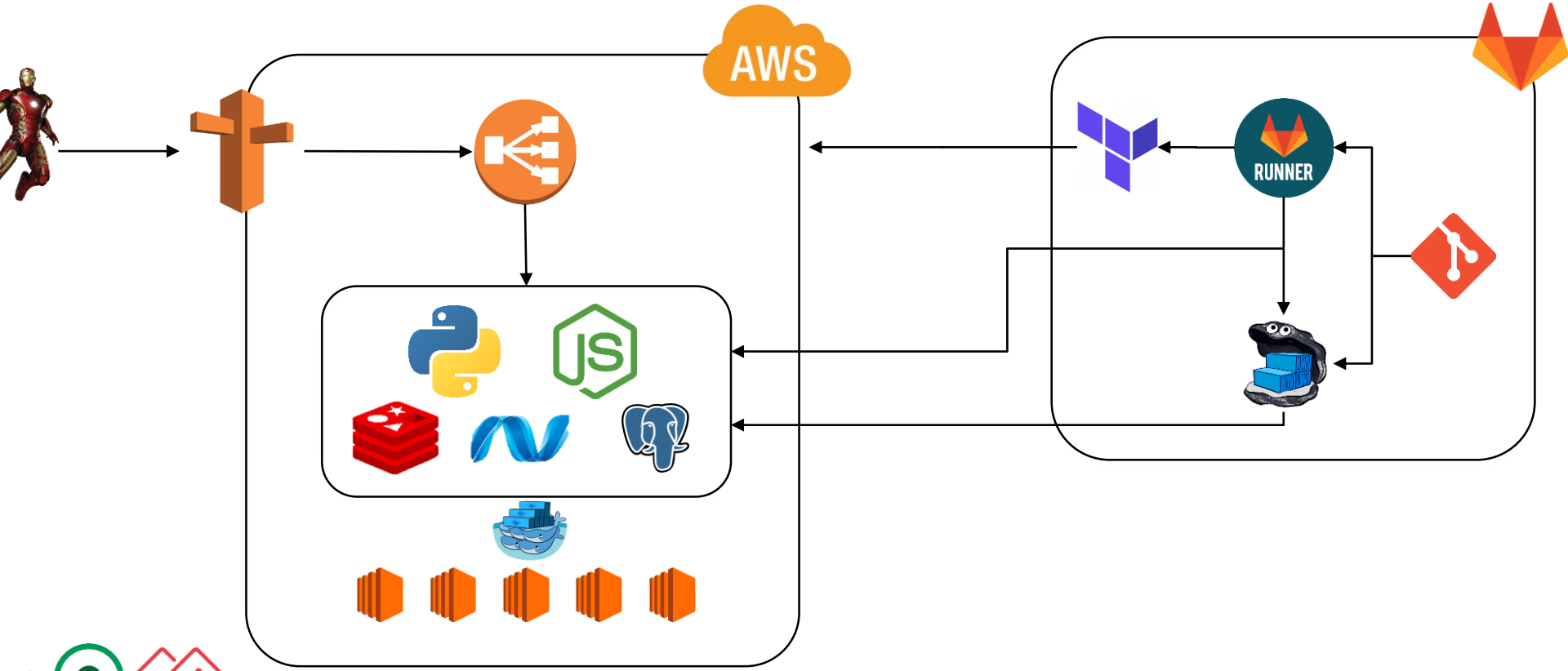
This is DareDeveloper, Welcome to Full Auto CD


Artifact Repository
fetch
VCS
source
push
Execution Environment
AWS
"run!"
"CI" Service
build & test
RUNNER

Demo

Gitlab
Service Config

Big Picture :tm:



Deployments

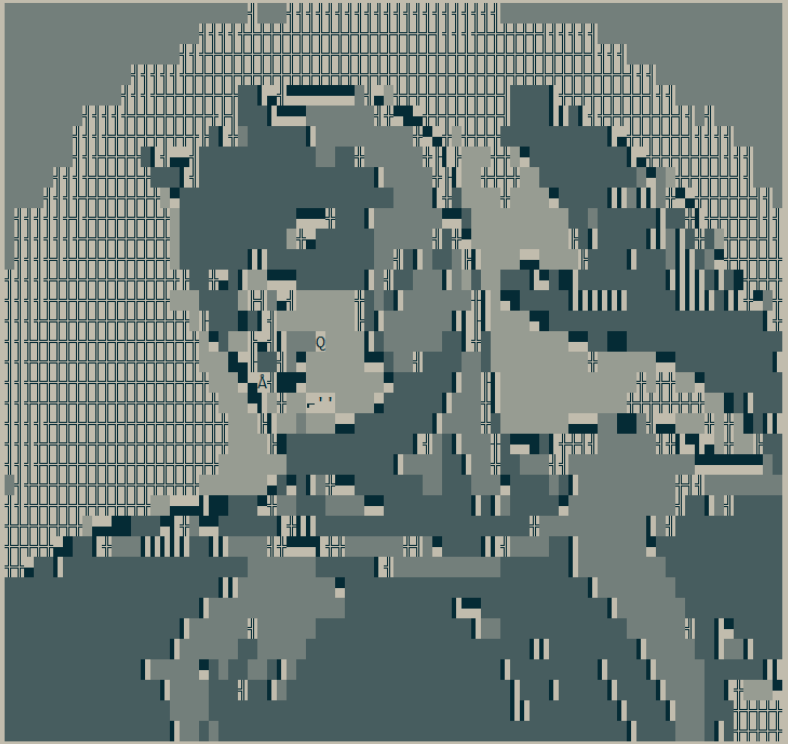
- Deployment Strategies
  - Rolling Deployment
  - Green/Blue Deployment
  - Canary Deployment

- Feedback from Logging & Metrics

Docker Build

- Gitlab Runner running in a Docker container
- Dedicated Docker container for each job in the pipeline
- By distributing the runner „swarm-scale“ build cluster can be formed

```
# cat /home/ubuntu/.daredeveloper/secret_diary
```



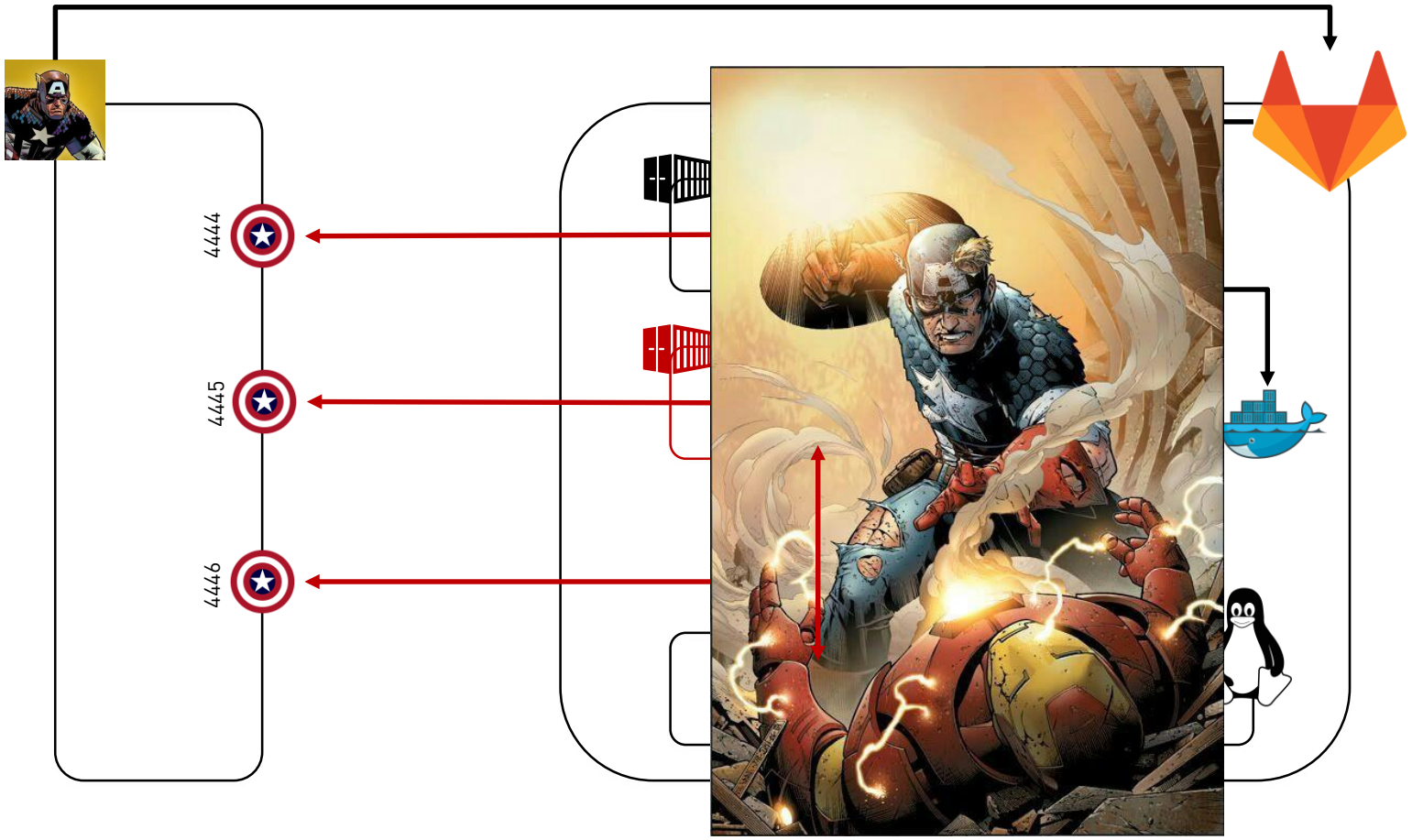
I'm blind, people start to notice

WARNING

The Following stunts are performed by idiots or under the supervision of stupider idiots. If you attempt to reenact any of the idiotic stunts you will see, you will be left with idiotic scars.

Enjoy.

This is Captain Security, Welcome to Swarm Pwnage!


4444
4445
4446

So Far: Build System Breakout

- We looked at one particular aspect
- Raising awareness for capabilities of the Docker socket
  - Good summary: blog.heroku.com

More Security Challenges

- Build System Breakout
  - Traversing to Production Swarm
- Breakout on Production Swarm
- Deployment of Vulnerable Code to Production
- Registry Overwrite
- Leaking of Secrets

Build System Breakout

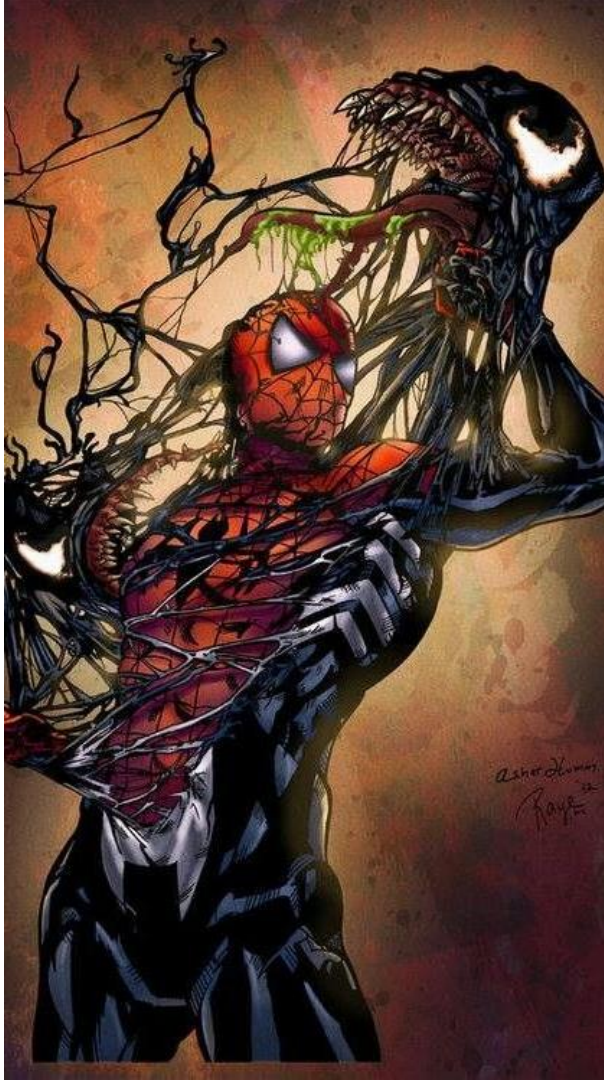
- Inherent problem of the requirement to build docker containers in build containers
  - Only configuration-based review of drone.io, Travis, and Jenkins, but most likely the same issues
- Potential Solution:
  - Verify Build Scripts
  - Build System Zoning
  - Docker Image Build Without Docker

Docker Image Build Without Docker

- Several projects available
  - https://github.com/genuinetools/img
  - https://bazel.build/
  - https://github.com/projectatomic/buildah
- Need to build a custom build image/runner/plugin

Build System Breakout – Lateral Movement

- Common approach: Build system deploys to production platform
- If so:
  - Credentials can be accessed after breakout
  - Lateral Movement to target systems



Breakout on Production Swarm

- Container breakout vulnerabilities are relevant
- Cluster escalation vulnerabilities are relevant
- Not the scope here!
  - Refer to H2HC 2017 – Pentesting DevOps

Breakout on Production Swarm

- In scope: Deploying privileged container
  - Docker: `privileged: true` in compose file
  - Kubernetes: `privileged` flag in deployment yml
- No proper way to restrict container runtime options on a cluster level

DoS on Production Swarm

- Binding container to relevant (host) port
- `docker service create -p 22:22 IMAGE …`

- Potential Solution for both DoS/Breakout:
  - Deployment Verification

Deployment of Vulnerable Images

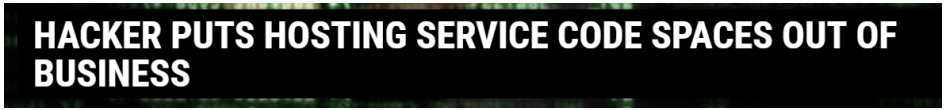
- Probably the most popular “Secure Pipeline” aspect
- Possibility has existed longer than containers/DevOps/CD
  - Still a very important aspect

Deployment of Vulnerable Images

- Challenge: Scan all of
  - OS packages
  - App code
  - App dependencies
- But again:
  - Build Verification Required

Registry Overwrite

- Build System must be able to push to registry
- Broad registry push access allows to overwrite images
- => Review your registry user/role/permission concept!



Leaking of Secrets

- Builds (and thus repositories) must contain secrets on a regular basis
- For example:
  - Registry login credentials
  - Credentials for application stack
- Secrets in a repository are a bad idea ;-)

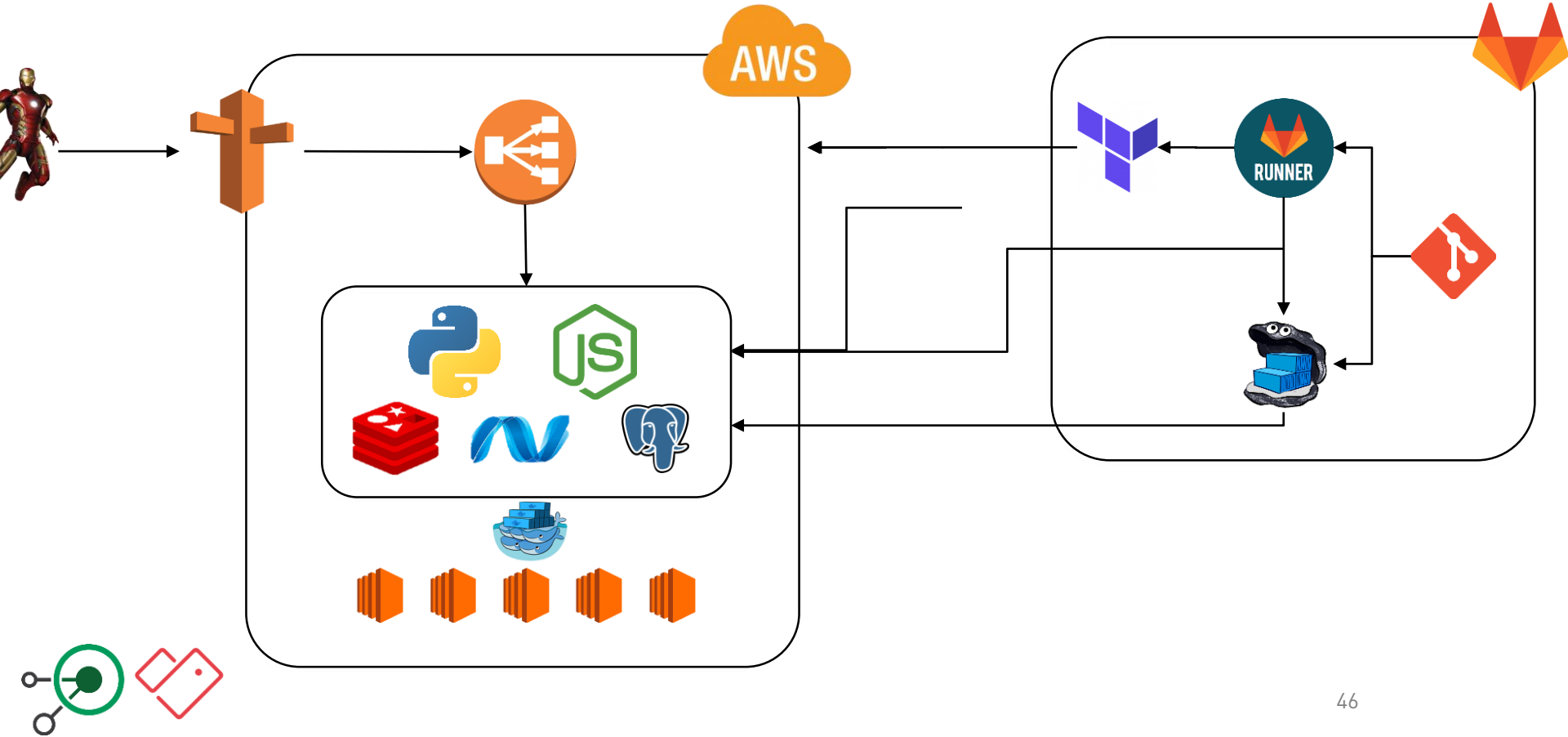
Secret Injection

- All major build systems support secret injection of some kind
  - `docker login -u gitlab-ci-token -p $CI_BUILD_TOKEN $DOCKER_REGISTRY`
- Ensure usage!
  - Scan repos for secret storage
    - Various tools available
  - Support developers
    - Awareness
    - Provide `.gitignore`/commit hook configs

Validate Builds & Deployments

- We saw the need for validation to address
  - Build Breakout
  - App/Image Vulnerabilities
  - Deployment Mal-/Mis-Configuration
- How to enforce validation?
- Build Systems don't provide mandatory check steps without "manual work"...

Architecture Overview



What do we want to scan for?

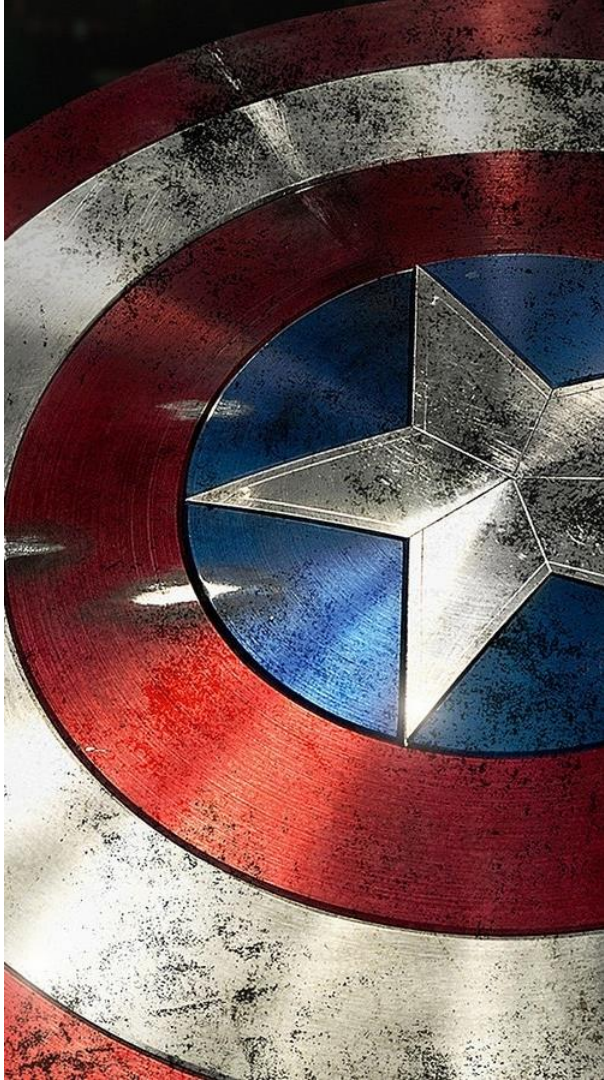
- Application Vulnerabilities
- OS Package and Library Vulnerabilities
- Build Script Breakout
  - As always, hard to detect.
  - Docker-less builds essential
- Deployment Mal-/Mis-Configuration

What do we want to scan for?

- Deployment Mal-/Mis-Configuration
  - Capabilities
  - AppArmor/seccomp profile deactivation
  - Container running as root
  - Secrets
  - Network policies
  - Namespace sharing
  - Mount propagation/Volumes
- See CIS Docker/K8s benchmarks as well.

Tooling?

- Some “Container Security” tools/scanners start to provide the above.
  - … some really do not.
- No tool recommendation here, no extensive technical evaluation performed.
- The slides above provide your evaluation checklist



Security Zoning

- Captain Security's favorite approach:
  - Use dedicated environments per application project/stack/team.
  - Basically remove multi-tenancy from the vulnerability list
- We saw above how easy it is to deploy the complete infrastructure.
- Granularity of zoning depends on your environment.
  - E.g. according to business unites, protection need/classification, ...

Network Isolation

- Old controls still relevant in the old world
- Build system/production swarm only needs filtered outbound network access
  - Proxy-based whitelisting
    - Very feasible!
- Of course no prevention, but raising the bar.

Empower & Collaborate

- As a security team, provide
  - Training in the new CD approaches
  - Repository scaffolding
    - Commit hooks including checks for included secrets
    - `.gitignore`
    - `.gitlab-ci.yml` including all scanning checks
      - Provide scanning checks as well ;-)
  - Container security policies (e.g. K8s Pod Security Policies) available in the cluster

Empower & Collaborate

- If implemented properly and securely:
  - Immutable infrastructures are immutable
    - And easy to baseline!
  - Executable and version-able infrastructure/design documentation
  - Automated security checks
  - Timely patching
  - Centralized secret management

Summary

- Practical demo of a functional CD pipeline
  - Including war stories from production
  - Not just some marketing slides of distant blog posts from some .io startup.
- Illustration of multi-tenancy issues
- Limitations of build systems when it comes to mandatory checks

Summary

- Focus on technical aspects
  - Short touch on awareness/motivation via belts/guilds
- Extending CD security aspects beyond “integrate a source code scanner”
- CD provides awesome possibilities to improve security!

Questions?

Thank you for your attention!

barth@stocard.de
mluft@ernw.de

@der_cthulhu
@uchi_mata

www.stocard.de
www.ernw.de

www.insinuator.net

Shoutouts

- Simon who built a lot of basic infrastructure that was (re-) used in this talk!

Disclaimer

- All trademarks, product names, company name, publisher name and their respective logos/images/media cited herein are the property of their respective owners.