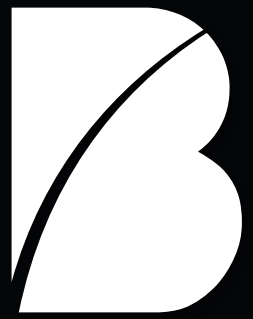


Securing the Automated Pipeline: A Tale of Navigating the Microservices Culture

Julie Chickillo – VP, Information Security

Brandon Grady – SVP, Software Engineer

Ben Finke – Threat and Vulnerability Program Manager

beeline®

ABOUT US

Locations

- Headquarters:
  - Jacksonville, FL
- Offices:
  - Denver, CO
  - Australia, Melbourne
  - Philippines, Manila
  - United Kingdom, London
  - Rochester, MI

36B+

ANNUAL SPEND UNDER MANAGEMENT

40% of spend is SOW / Services Procurement

550+

EMPLOYEES

With more than 1,600 years of industry experience

300+

CUSTOMERS

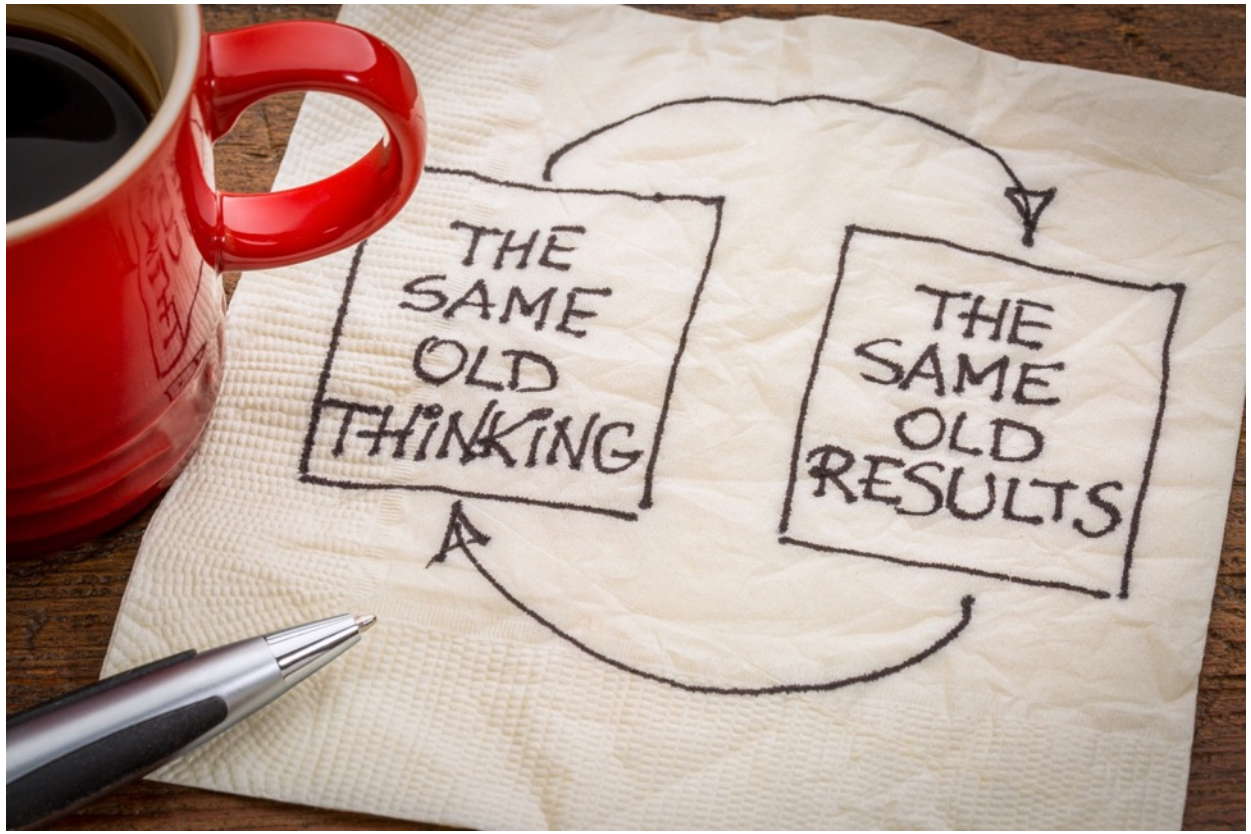
30%+ are Fortune 500 / Fortune Global 500

Back in My Day . . . . .

We rented a place in a colo, bought a server, installed an OS on it, brought up a web server, and once a quarter we deployed code on it.

Deploying Software - uphill both ways!

1. Code is developed
2. Code is placed in UAT
3. Dynamic and Static Scanning
4. Tickets entered for vulns
5. Devs fix and redeploy
6. Code retested
7. Functional Testing
8. Manual App Pentesting
9. Code Push to Prod scheduled
10. Whole weekend wasted as production push fails and dev and infrastructure teams blame each other, finally get the stupid thing working in time for the Monday morning standup.



Standard Security Tools for the Good Ole Days…

- Static Source Code Analysis = 8 hours
- Dynamic Scan Analysis = 2 hours
- Vulnerability Scans = 1 hour
- Configuration Management – Agent Software or Network Scanner
- Attack Detection – Agent software and Network Device
- Web App Firewall – 1-2 weeks of tuning in UAT

Then This Happened

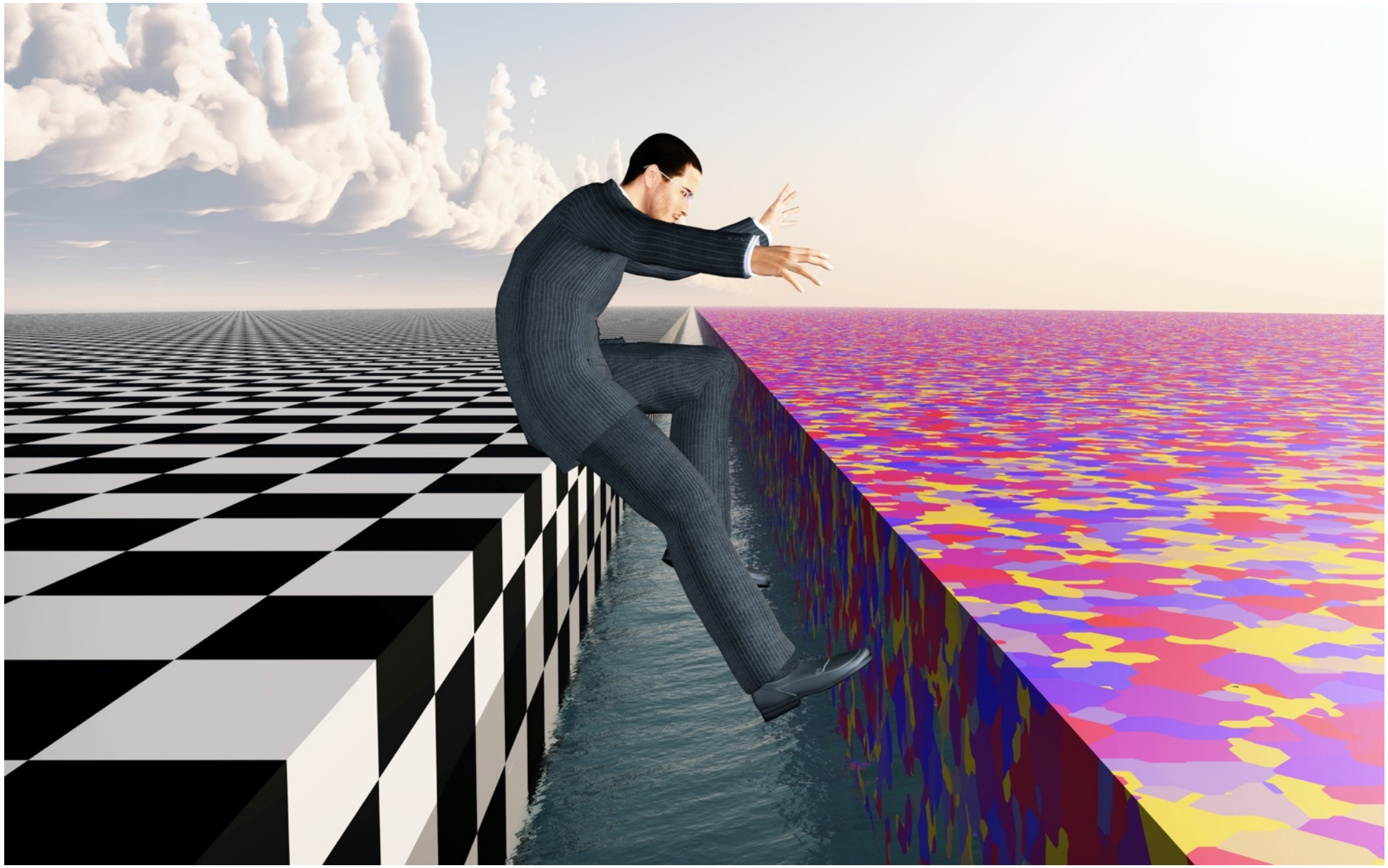
Otherwise known as the time security said “You built What!!!!”

The Speed of DevOps

- Continuous Integration and Infrastructure as Code means an environment can be fully deployed in 15 minutes.
- Containers spin up and down automatically
- Able to deploy small, durable changes multiple times per day.




Paradigm Shift



Continuous Integration Deployment Cycle

1. Code Development and development QA
2. Branch merged into MASTER
3. Artifact built and deployed to ALL Dev Envs - containers spun up, code deployed, services started
4. Functional Testing Executed – artifact deployed to QA
5. Successful testing, branch is tagged
6. Artifact deployed to PROD

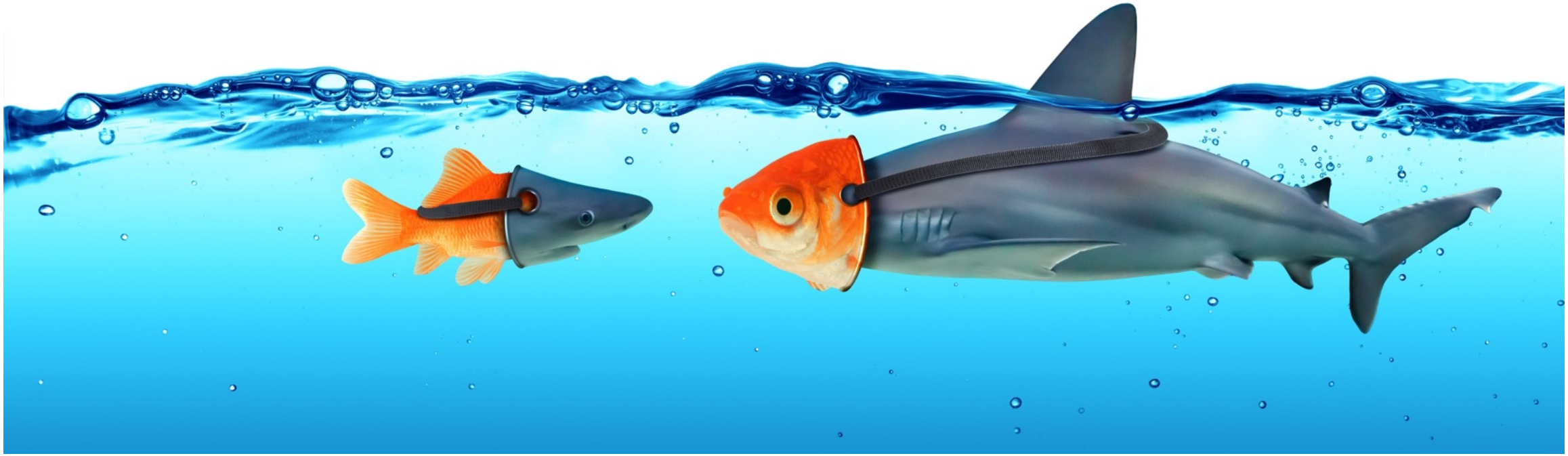
Brandon tried to make this slide way more complicated, by the way…

Change of Mindset

Standard Security Tools for the Good Ole Days…

- Static Source Code Analysis = 3 hours
- Dynamic Scan Analysis = 2 hours
- Vulnerability Scans = 1 hour
- Configuration Management – Endpoint Agent Based
- Attack Detection – Agent software and Network Device
- Web App Firewall – 1-2 weeks of tuning in UAT

Building trust



Starting the conversation

Giving Devs the power to say no!

A little thing I like to call . . Wait for it . . . . .

Talking with the Dev Teams!
GASP Say it isn't so!

Continuing the conversation – yes you do have to keep talking with the developers.

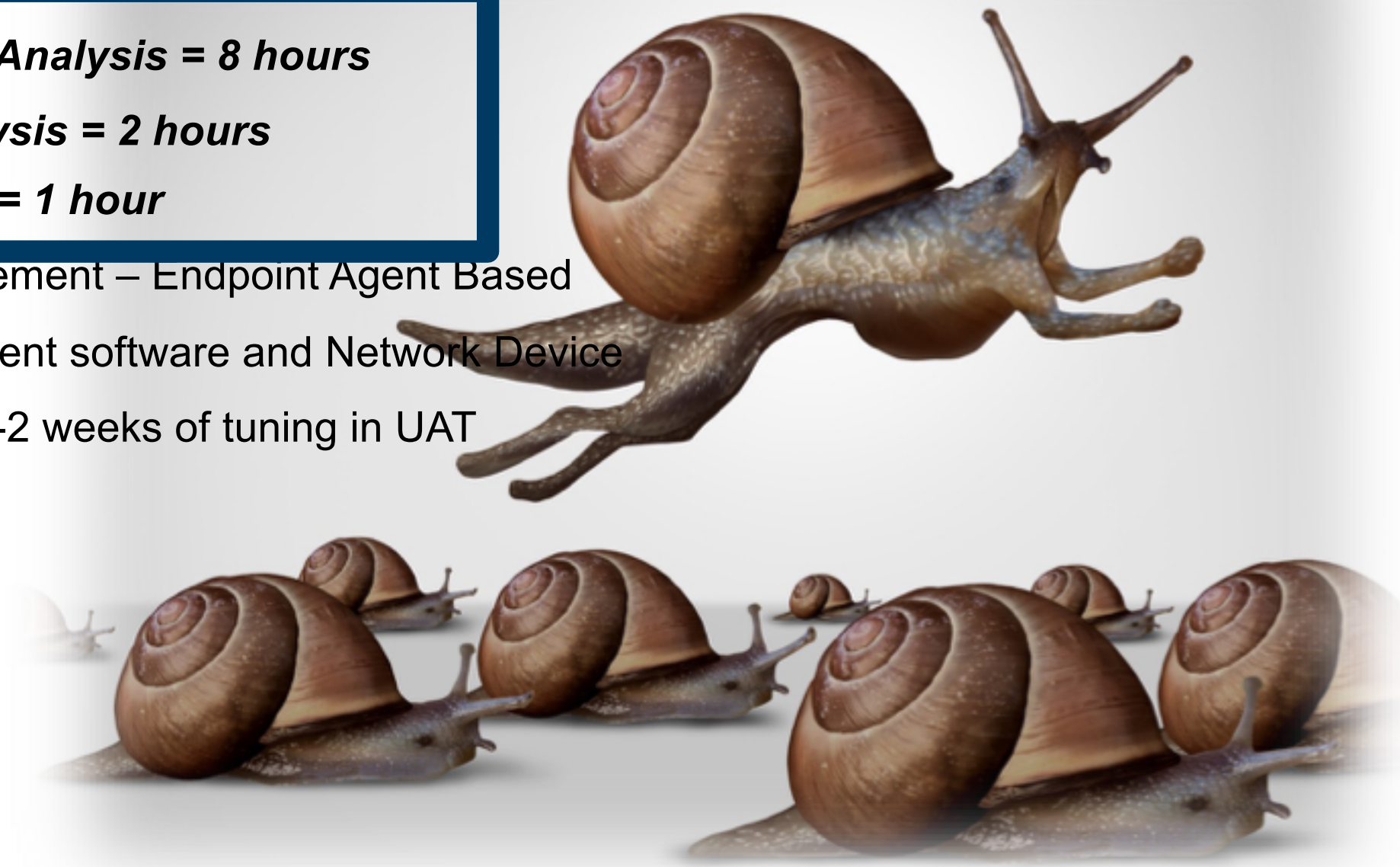
OK, Catching Our Breath, New Security Requirements

1. **Must be able to match deployment speeds**
2. Must be able to work in multiple environments (containers, ugh)
3. Provide certainty that production code is clean
4. Keep the devs happy
5. Keep the auditors happy
6. Keep the security team happy!*

**Self Assigned Goal…*

1. Match the Speed

- ***Static Source Code Analysis = 8 hours***
- ***Dynamic Scan Analysis = 2 hours***
- ***Vulnerability Scans = 1 hour***
- Configuration Management – Endpoint Agent Based
- Attack Detection – Agent software and Network Device
- Web App Firewall – 1-2 weeks of tuning in UAT



Automated Reviews at Speed

- Standard Static and Dynamic tools too slow
- Interactive Testing – lets us piggyback the functional testing gates
- Needed a tool that could run in both old school (on-prem), Cloud, Hybrid & DevOps worlds
- Can handle Windows and Linux, Containers and Servers, .NET/Java/Node.js
- Runs in minutes on AWS

What Beeline is using today

OK, Catching Our Breath, New Security Requirements

1. Must be able to match deployment speeds
2. **Must be able to work in multiple environments (containers, ugh)**
3. Provide certainty that production code is clean
4. Keep the devs happy
5. Keep the auditors happy
6. Keep the security team happy!*

**Self Assigned Goal…*

1. Match the Speed

- Static Source Code Analysis = 8 hours
- Dynamic Scan Analysis = 2 hours
- ***Vulnerability Scans = 1 hour***
- ***Configuration Management – Endpoint Agent Based***
- ***Attack Detection – Agent software and Network Device***
- Web App Firewall – 1-2 weeks of tuning in UAT

2. Runs wherever the code is

- IAST tool runs as a component of the app, deployed as part of the underlying app server
  - Containers
  - Virtual Servers
  - PaaS Platforms
- Build into the deployment process
  - Deploy software and configurations consistently
  - Include security concerns into health check functions

```
# Pull base image.
FROM ubuntu:14.04

# dependencies
RUN apt-get update && apt-get -y install build-essential curl git-core libcurl4-openssl-dev \
libc6-dev libreadline-dev libssl-dev libxml2-dev libxslt1-dev libyaml-dev zlib1g-dev libssl-dev imagemagick

# Ruby install
RUN curl --progress http://cache.ruby-lang.org/pub/ruby/2.3/ruby-2.3.0.tar.gz | tar xz && \
  cd ruby-2.3.0 && ./configure --disable-install-doc && \
  make && make install && cd .. && rm -rf ruby-2.3.0* && \
  echo 'gem: --no-document' > /usr/local/etc/gemrc && \
  gem install bundler

# SqLite
RUN apt-get -y install sqlite3 libsqlite3-dev

# Set locale
RUN locale-gen en_US.UTF-8
ENV LANG en_US.UTF-8
ENV LANGUAGE en_US:en
ENV LC_ALL en_US.UTF-8
```

2. Runs wherever the code is

Verifying the integrity of the underlying server/container

1. Established a trusted registry for images
2. Only allow deployments of accepted containers/software
3. Script up checks for changes to registries (easier than it sounds!)
4. Supply Chain concerns go here…
5. Configuration checking happens here too
6. Complete software component inventory!!!!!! (IMPORTANT)

OK, Catching Our Breath, New Security Requirements

1. Must be able to match deployment speeds

2. Must be able to work in multiple environments (containers, ugh)
3. **Provide certainty that production code is clean**
4. Keep the devs happy
5. Keep the auditors happy
6. Keep the security team happy!*

**Self Assigned Goal…*

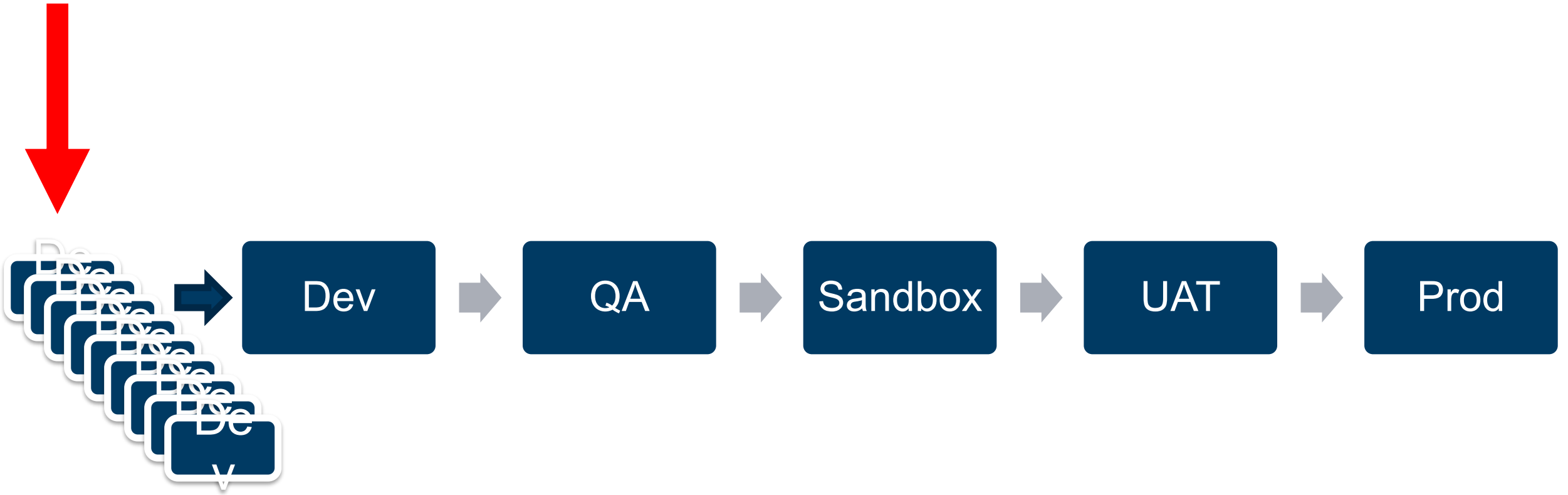
Phase 1 - Gatekeeper Approach

Dev → QA → UAT → Prod

Phase 2 - Pushing Testing to the Left

Still doing the Gatekeeper process

Phase 3 – Individual Dev Testing



Still doing the Gatekeeper and QA/DEV Testing

OK, Catching Our Breath, New Security Requirements

1. Must be able to match deployment speeds

2. Must be able to work in multiple environments (containers, ugh)

3. Provide certainty that production code is clean
4. **Keep the devs happy**
5. Keep the auditors happy
6. Keep the security team happy!*

**Self Assigned Goal…*

4. Keeps the devs happy

App Security is just another part of QA.

- Puts any findings into the exact same workflow as any bug ticket
- Provide full context and details in the same ticket
- Provide priority and significance of issues
- Give them an opportunity to help build the process

Unhappy devs will find always find ways around your security process

OK, Catching Our Breath, New Security Requirements

1. Must be able to match deployment speeds
2. Must be able to work in multiple environments (containers, ugh)
3. Provide certainty that production code is clean
4. Keep the devs happy
5. **Keep the auditors happy**
6. Keep the security team happy!*

**Self Assigned Goal…*

5. Keep the auditors happy

Documenting the process

- Approval of code for prod
- Jira ticket created for every build
  - Add webhook to CI server to query Contrast API after functional testing, record output in ticket
- Use version-commit numbers in your testing
- Remember – no findings is not the same as proof it ran

OK, Catching Our Breath, New Security Requirements

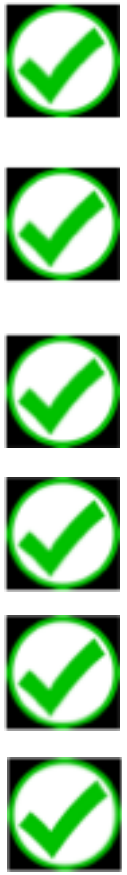
1. Must be able to match deployment speeds
2. Must be able to work in multiple environments (containers, ugh)
3. Provide certainty that production code is clean
4. Keep the devs happy
5. Keep the auditors happy
6. **Keep the security team happy!***

**Self Assigned Goal…*

6. Keep the security team happy!

- Automate wherever possible
- Security doesn't mean "No"
- Share experiences with other infosec peers

OK, Catching Our Breath, New Security Requirements



1. Must be able to match deployment speeds
2. Must be able to work in multiple environments (containers, ugh)
3. Provide certainty that production code is clean
4. Keep the devs happy
5. Keep the auditors happy
6. Keep the security team happy!*

**Self Assigned Goal…*

CI Deployment Cycle – Now with Security!

1. Code Development
2. Branch merged into QA
3. QA branch deployed - containers spun up, code deployed, services started, interactive security testing in place
4. Functional and Security Testing Executed
5. Successful testing, branch is tagged
6. Production Deployment – only tagged commits allowed

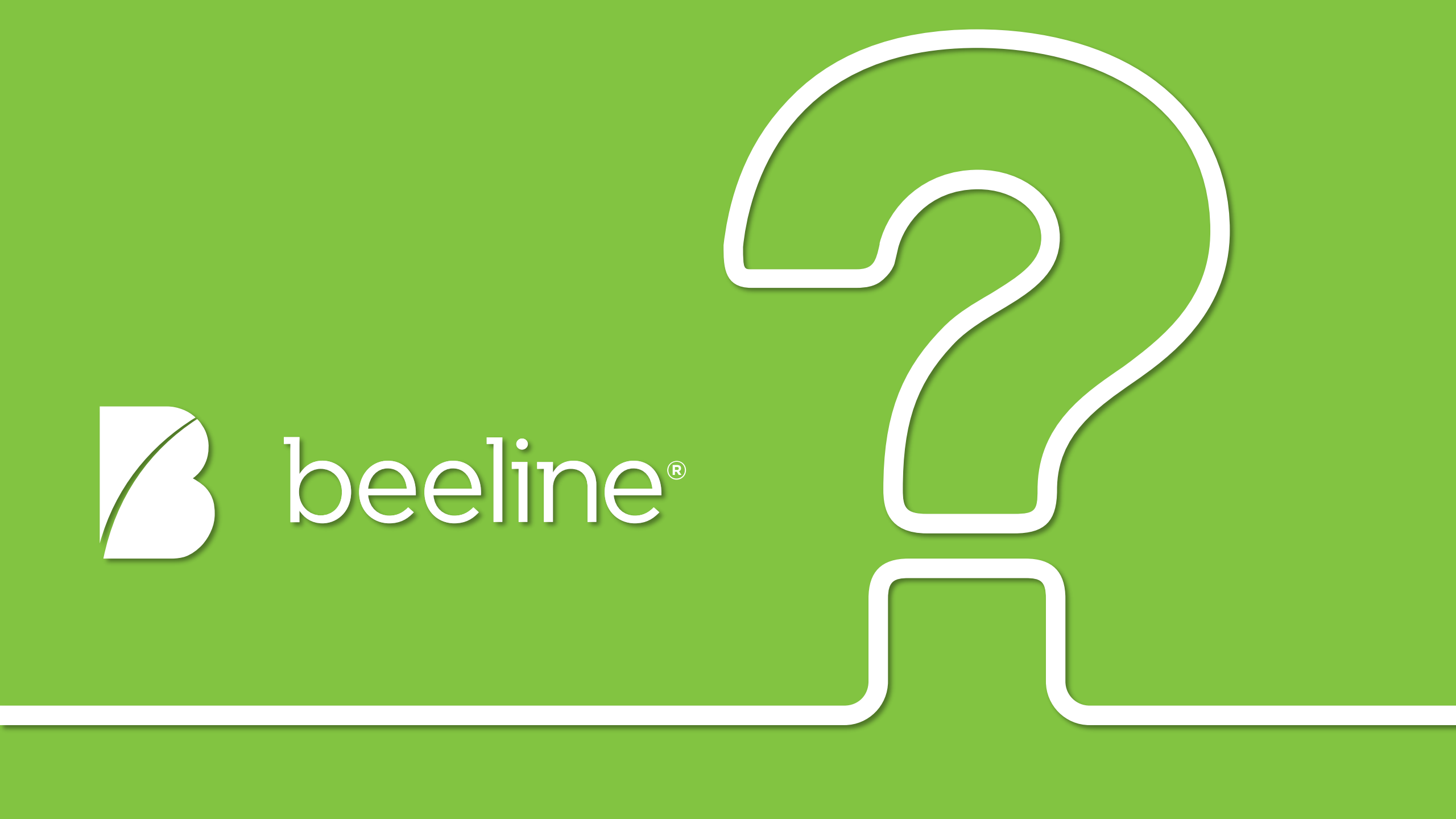
Skillz for Infosec peeps in the DevOps Age

- Learn to read (and write a little) code
- Understand containers
- Learn about continuous integration
- Infrastructure as code is your friend
- Infosec is the natural bridge between dev and infra teams

CALL TO ACTION

- Availability of **Contrast Community Edition** running on AWS
- **FREE** Community Edition that help protect web applications and APIs running on AWS from cyber attacks
- Empowers development and security teams to improve security (of both custom code and third-party components) for cloud migrations to AWS and applications already running on AWS
- Delivers the power of Contrast Protect and Contrast Assess on AWS

https://www.contrastsecurity.com/contrast-community-edition


beeline®

Contact Information

security@beeline.com

https://www.beeline.com/resources/vendor-management-system-vms/

THANK YOU