goto;
chicago

Click ‘Rate Session’

Rate **5** sessions to get the supercool GOTO reward

follow us @gotochgo

OpenCredo
A TRIFORK COMPANY

JOURNEYS TO CLOUD NATIVE ARCHITECTURES

SUN, SEA & EMERGENCIES

Nicki Watt - CTO
@techiewatt

How do you learn?

How do you decide what works, and what doesn’t?

Feeling pain
Learning from failures

JOURNEYS TO CLOUD NATIVE ARCHITECTURES

SUN, SEA & EMERGENCIES

@techiewatt

JOURNEYS TO CLOUD NATIVE ARCHITECTURES

SUN, SEA & EMERGENCIES

@techiewatt

ABOUT ME / OPENCREDO

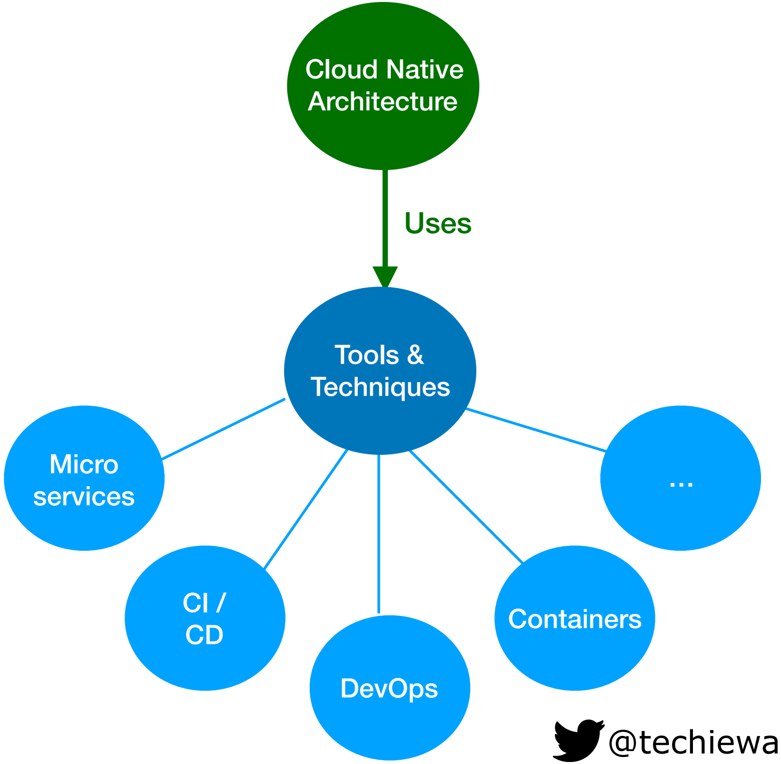
- CTO at OpenCredo
- Hands on software development consultancy
- Cloud Native; Data Engineering & ML

AGENDA
OVERVIEW

- OVERVIEW - THE DREAM
- ISSUES & CHALLENGES
- CONCLUSION

Overview – The Dream

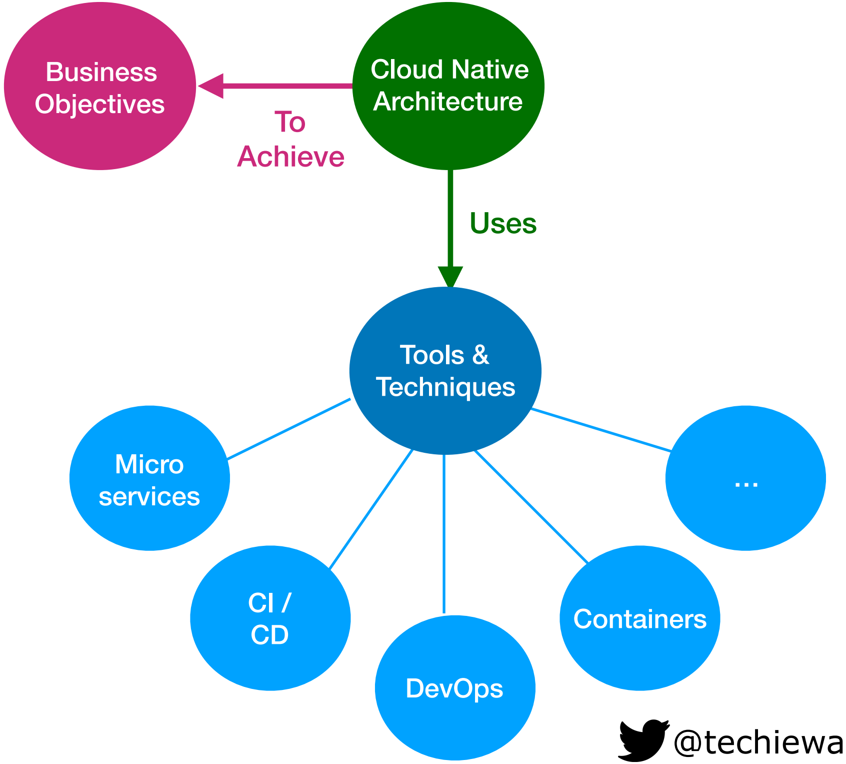
THE CLOUD NATIVE DREAM (BUSINESS DRIVERS)



@techiewatt

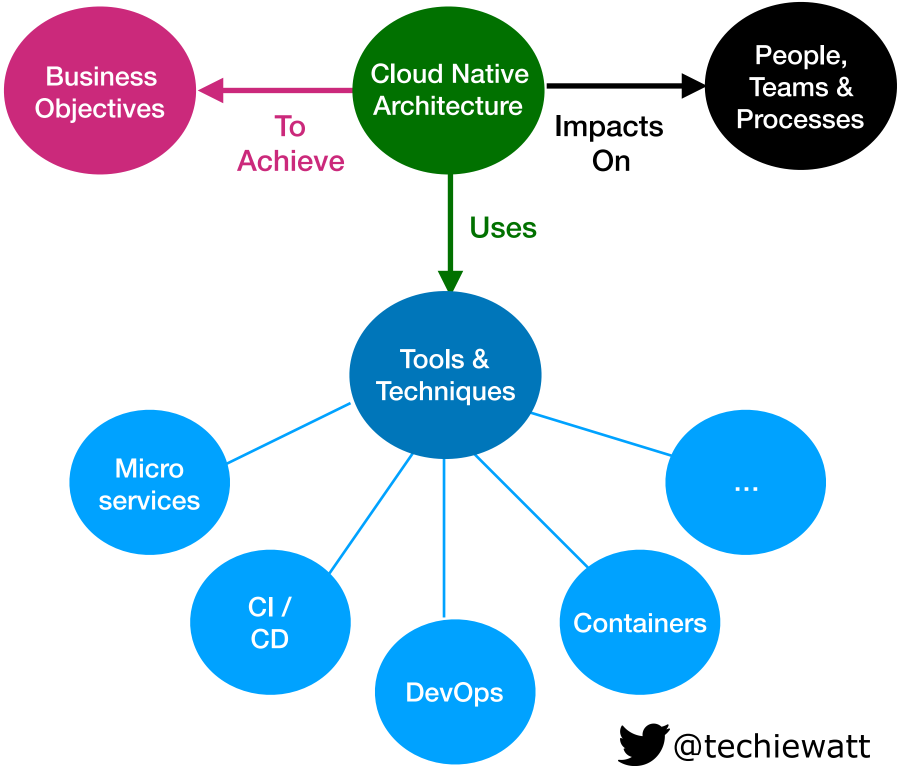
THE CLOUD NATIVE DREAM (BUSINESS DRIVERS)

- Faster Time to Market
- Allow business to grow (Scalability)



THE CLOUD NATIVE DREAM (BUSINESS DRIVERS)

- Faster Time to Market
- Allow business to grow (Scalability)



@techiewatt

"Going Cloud-Native" is an approach to build and run applications capable of exploiting and taking advantage of modern, scalable cloud like delivery techniques and tools

in order to meet business objectives!

Check In . . .

Underlying Challenge:

Too Much Baggage

Photo by Laura Gilmore (Flickr)

@techiewatt

THE CLOUD NATIVE JOURNEY

- Needs A Different
  - Approach
  - Tooling
  - Skills

Photo by Laura Gilmore (Flickr)

@techiewatt

Departure . . .
Arrival . . .
The Holiday Begins . . .

Issues & Challenges

#1
The Jones'

#1
Tackling Too Much Too Quickly

(Sunburn)

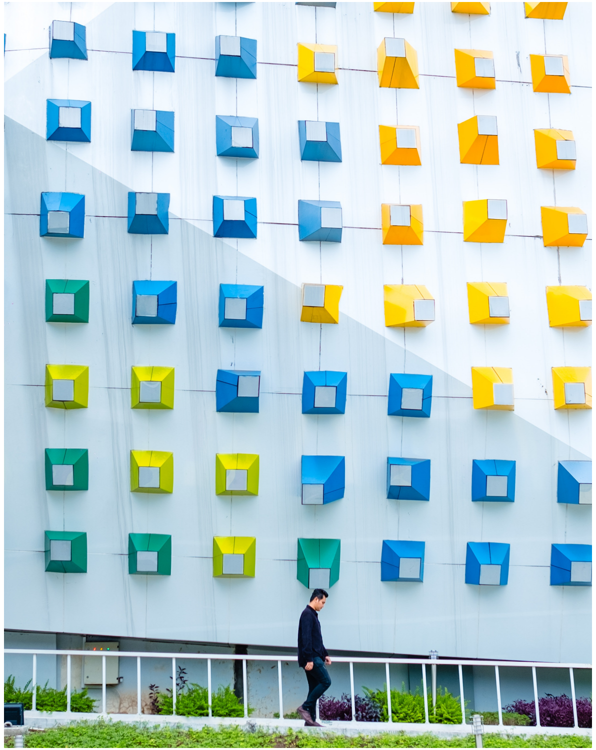
TACKLING TOO MUCH, TOO QUICKLY

Can't expect to gain a
fully rounded & reliable
cloud native system overnight!

@techiewatt

TACKLING TOO MUCH, TOO QUICKLY

- Many Moving Parts
  - App Architecture
  - Ops
  - Infrastructure



@techiewatt

TACKLING TOO MUCH, TOO QUICKLY – OBSERVATIONS

- Quality is Compromised
- Pain is delayed - focusing on easy not risky
- Operate under a false sense of security

TACKLING TOO MUCH, TOO QUICKLY - OBSERVATIONS

- Quality is Compromised
- Pain is delayed - focusing on easy not risky
- Operate under a false sense of security

image credit: http://www.guinnessworldrecords.com/world-records/most-tennis-balls-held-in-the-mouth-dog

@techiewatt

TACKLING TOO MUCH, TOO QUICKLY – OBSERVATIONS

- Quality is Compromised
- Pain is delayed - focusing on easy not risky
- Operate under a false sense of security
    - Only happy path testing (platform testing neglected)

@techiewatt

TACKLING TOO MUCH, TOO QUICKLY – OBSERVATIONS

- Quality is Compromised
- Pain is delayed - focusing on easy not risky
- Operate under a false sense of security
    - Only happy path testing (platform testing neglected)
    - Lack of robust CI/CD - Inconsistent environments

TACKLING TOO MUCH, TOO QUICKLY

Slow Down

Take Baby Steps

Focus On Quality!

#2
Deepak and friends

#2
Relying on a Surface Level Understanding

(Caught In Deep Water)

@techiewatt

RELYING ON A SURFACE LEVEL UNDERSTANDING

Distributed Systems are hard.
The devil is in the detail
And the detail matters

RELYING ON A SURFACE LEVEL UNDERSTANDING

- **Skills & Real World Experience**
  - Kubernetes in 24 hours != Expert
  - Outsource commodity, in-house business value

RELYING ON A SURFACE LEVEL UNDERSTANDING

- **Skills & Real World Experience**
  - Kubernetes in 24 hours != Expert
  - Outsource commodity, in-house business value
- **Missing out on optimising for the Bigger Picture**
  - Artificial Boundaries

RELYING ON A SURFACE LEVEL UNDERSTANDING

- **Skills & Real World Experience**
    - Kubernetes in 24 hours != Expert
    - Outsource commodity, in-house business value
- **Missing out on optimising for the Bigger Picture**
    - Artificial Boundaries
    - Artificial Centralisation

RELYING ON A SURFACE LEVEL UNDERSTANDING

- **Other War Stories …**
  - DRY Dependency Hell
  - The Singularity Fallacy

RELYING ON A SURFACE LEVEL UNDERSTANDING

Gain basic skills & expertise

Make commodity problems somebody else's

Learn by
Optimising for the bigger picture

#3
Susie & Jacob

#3
Ineffective Tools & Processes

(Between a rock and a hard place)

@techiewatt

Me

Simultaneously petrified & enjoying myself …

INEFFECTIVE TOOLS & PROCESSES

- Often traditional tools just don’t cut it

INEFFECTIVE TOOLS & PROCESSES

- **Mileage varies …**
  - Slow you down
  - Take you the long way round
  - Downright hurt you in long run
  - More Forward Looking

INEFFECTIVE TOOLS & PROCESSES

- **Slow you down**
    - Unnecessarily Heavy tooling (Some Enterprise Databases, Base Container Images)

INEFFECTIVE TOOLS & PROCESSES

- **Take you the long way around**
  - DIY or “Build Your Own” category

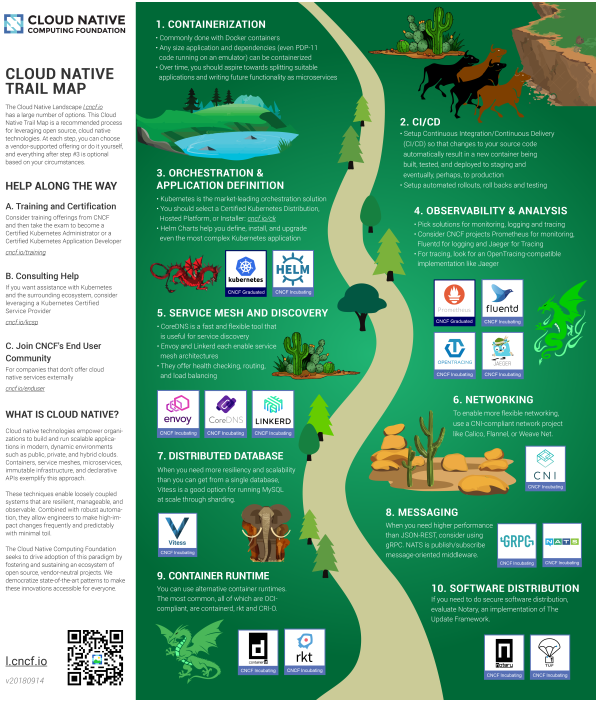
INEFFECTIVE TOOLS & PROCESSES

- **Downright hurt you**
    - No API
    - Non Automatable
    - Horizontally challenged

INEFFECTIVE TOOLS & PROCESSES

- **More forward looking …**
    - Imperative –> Declarative

INEFFECTIVE TOOLS & PROCESSES – SO WHAT SHOULD YOU USE?



CLOUD NATIVE COMPUTING FOUNDATION

CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape *l.cncf.io* has a large number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator or a Certified Kubernetes Application Developer

cncf.io/training

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider

cncf.io/kcsp

C. Join CNCF's End User Community

For companies that don't offer cloud native services externally

cncf.io/enduser

WHAT IS CLOUD NATIVE?

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone.

l.cncf.io

v20180914

1. CONTAINERIZATION

- Commonly done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices

2. CI/CD

- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing

3. ORCHESTRATION & APPLICATION DEFINITION

- Kubernetes is the market-leading orchestration solution
- You should select a Certified Kubernetes Distribution, Hosted Platform, or Installer: *cncf.io/ck*
- Helm Charts help you define, install, and upgrade even the most complex Kubernetes application

4. OBSERVABILITY & ANALYSIS

- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing-compatible implementation like Jaeger

5. SERVICE MESH AND DISCOVERY

- CoreDNS is a fast and flexible tool that is useful for service discovery
- Envoy and Linkerd each enable service mesh architectures
- They offer health checking, routing, and load balancing

6. NETWORKING

To enable more flexible networking, use a CNI-compliant network project like Calico, Flannel, or Weave Net.

7. DISTRIBUTED DATABASE

When you need more resiliency and scalability than you can get from a single database, Vitess is a good option for running MySQL at scale through sharding.

8. MESSAGING

When you need higher performance than JSON-REST, consider using gRPC. NATS is publish/subscribe message-oriented middleware.

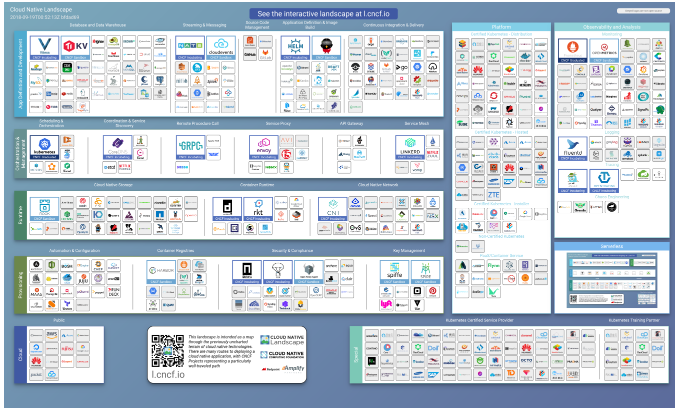
9. CONTAINER RUNTIME

You can use alternative container runtimes. The most common, all of which are OCI-compliant, are containerd, rkt and CRI-O.

10. SOFTWARE DISTRIBUTION

If you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.

Guidance from CNCF



@techiewatt

INEFFECTIVE TOOLS & PROCESSES - SO WHAT SHOULD YOU USE?

Considered Thought & Pragmatism …

(CNCF Recommendations - good start)

Rule of Thumb:

Automation, API & Horizontally scale friendly

#4
The Smiths'

#4 Ignoring Security

(Living Dangerously)

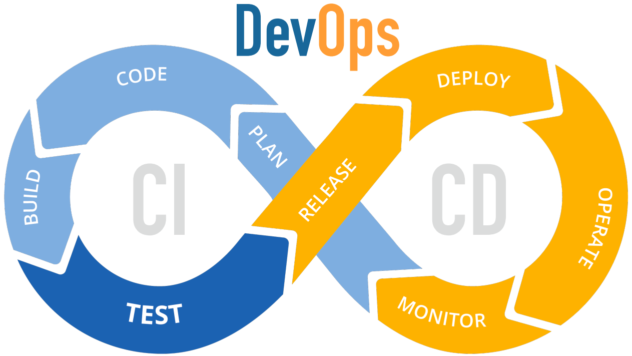
@techiewatt

IGNORING SECURITY CONCERNS

- **Enterprise Security has not kept up**

IGNORING SECURITY CONCERNS

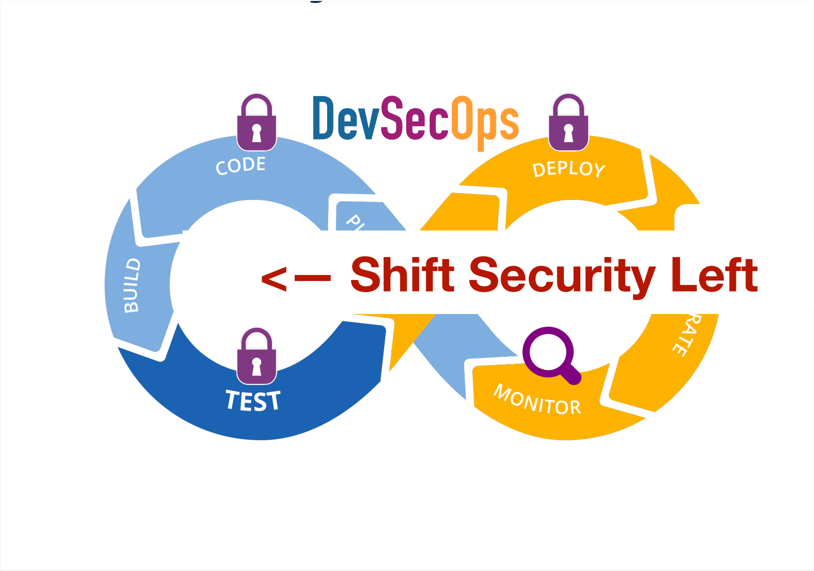
- **Enterprise Security has not kept up**
  - No excuse to ignore it though!
- **Devs/Ops are the new Gatekeepers**



@techiewatt

IGNORING SECURITY CONCERNS

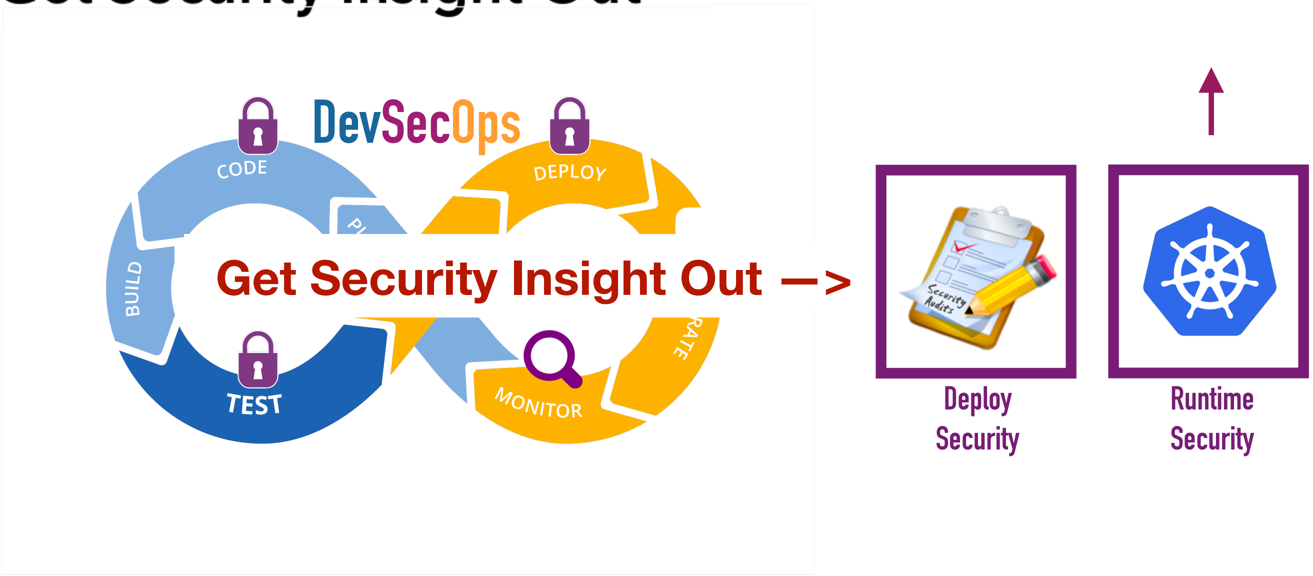
- **Shift Security Left**



<— Shift Security Left

IGNORING SECURITY CONCERNS

- Get Security Insight Out



Get Security Insight Out —>

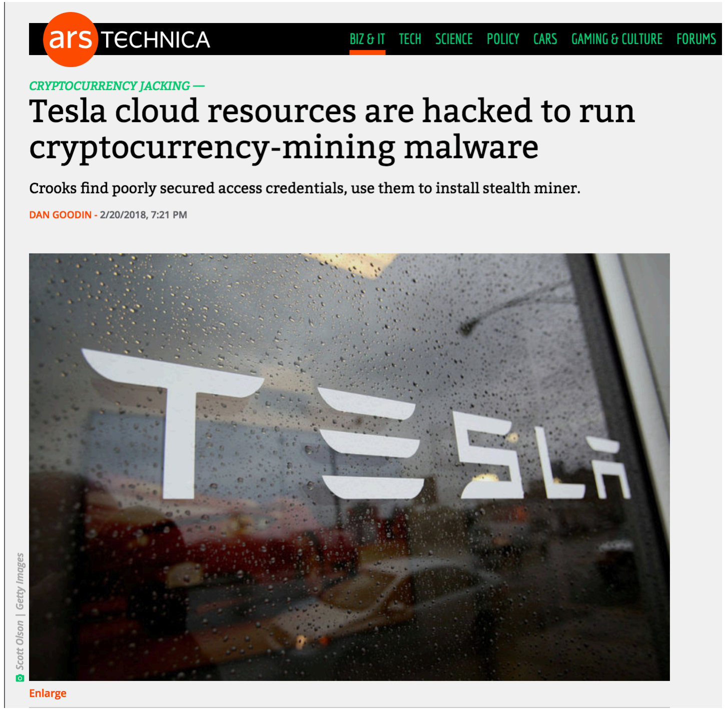
Deploy Security

Runtime Security

@techiewatt

IGNORING SECURITY CONCERNS

- Get Security Insight Out
  - Help a CISO, help yourself
  - Provide tools/reports to help understand new world



CRYPTOCURRENCY JACKING —

Tesla cloud resources are hacked to run cryptocurrency-mining malware

Crooks find poorly secured access credentials, use them to install stealth miner.

DAN GOODIN - 2/20/2018, 7:21 PM

Scott Olson | Getty Images

Enlarge

@techiewatt

IGNORING SECURITY CONCERNS

<—— Shift Security Left

Get Security Insights Out ——>

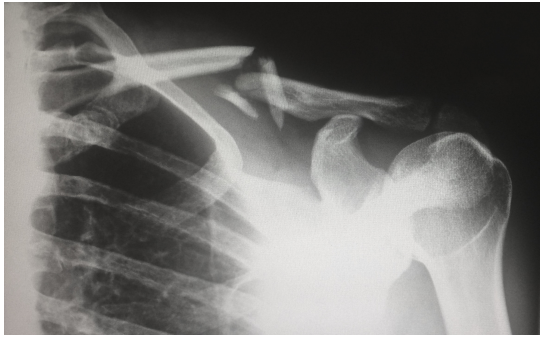
Engage don't enrage!

#5
Thandi

@techiewatt

#5
Not Anticipating Failure

(The Unexpected Accident)



@techiewatt

NOT ANTICIPATING FAILURE

- Test app & platform level sad paths

NOT ANTICIPATING FAILURE

- Test app & platform level sad paths
- Observability

NOT ANTICIPATING FAILURE

- Test app & platform level sad paths
- Observability
- First address unnecessary Chaos …
  then do … Chaos Engineering

NOT ANTICIPATING FAILURE

- Test app & platform level sad paths
- Observability
- First address unnecessary Chaos … then do … Chaos Engineering
- **Optimise for Recovery!**

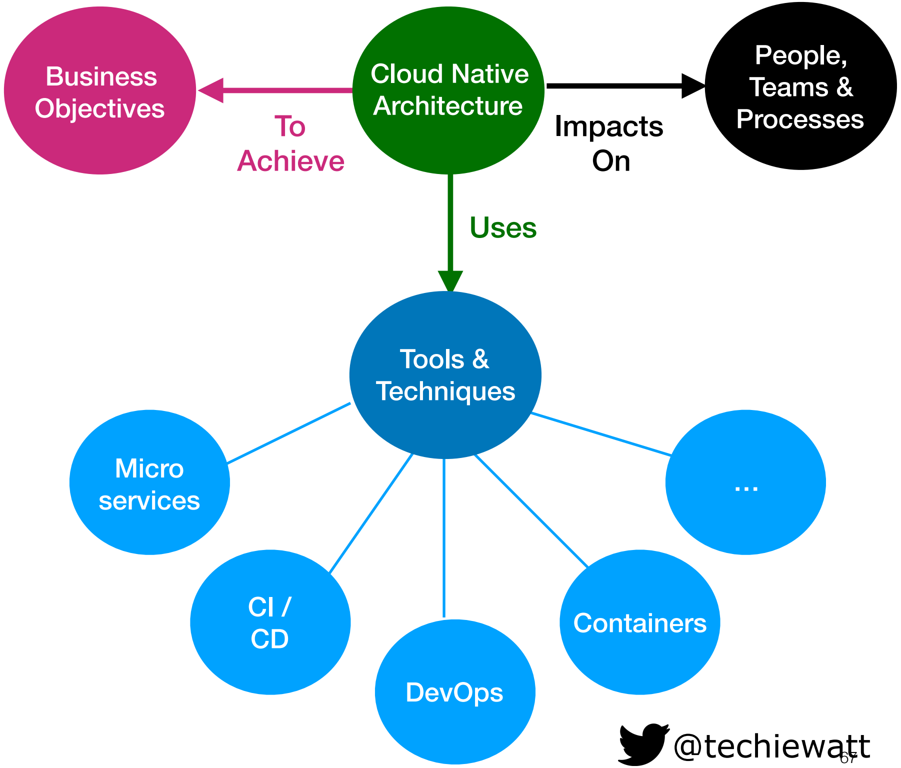
Failure is only really failure if you fail to learn from it

Summary & Principles for Success

@techiewatt

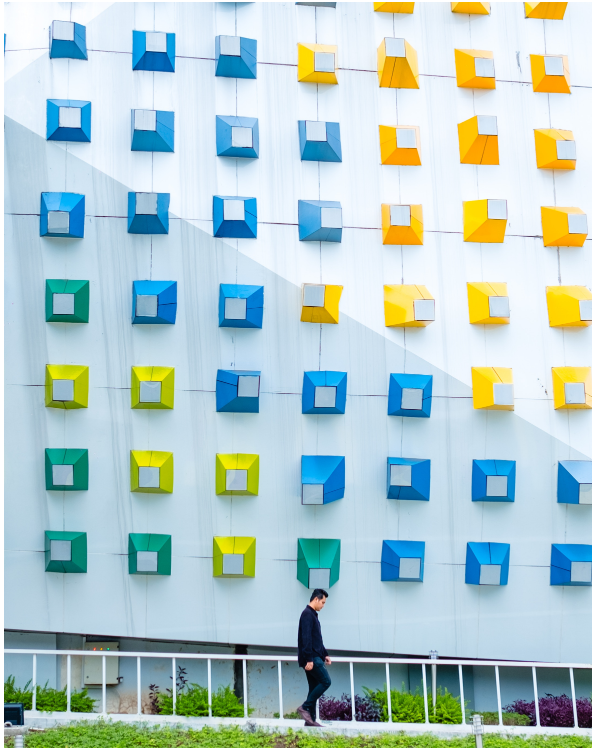
THE CLOUD NATIVE JOURNEY

- Needs A Different
  - Approach
  - Tooling
  - Skills



THE CLOUD NATIVE JOURNEY

- Needs To Cohesively Address:
  - App Architecture
  - Ops
  - Infrastructure



@techiewatt

PRINCIPLES FOR SUCCESS

- Don’t take on too much too quickly
- Acquire enough expertise to journey safely
- Choose your Tools wisely
- Engage don’t enrage Security
- Anticipate, Plan and Actively Test for Failure

Thanks!

Get in touch if you're looking for a partner to help on your Cloud Native Journey!

@techiewatt

goto; chicago

Click ‘Rate Session’

Rate **5** sessions to get the supercool GOTO reward

follow us @gotochgo