QCon NEW YORK

CLOUDZERO

ERIK PETERSON : @SILVEXIS : QCON NYC 2017

SERVERLESS SECURITY AND THINGS THAT GO BUMP IN THE NIGHT

HI, I'M ERIK

- Co-Founder, CEO, Engineer and Coffee Machine Technician @CloudZeroInc
- Reach me @Silvexis or erik@cloudzero.com
- I'm a recovering security, product and engineering person:
  - IT (UN IAEA, US Govt., SunTrust, Moody's), Software (Sanctum, GuardedNet, SPIDynamics, HP, Veracode)
- I'm now focused on Security as one attribute of Cloud computing and complex system design at CloudZero

CLOUDZERO

MORE DEV, EASY OPS, ALL SECURE

- **Our mission:** Help people build, secure and operate better Cloud applications with a focus on Serverless and Cloud Native architectures
- **We provide:** Radical Transparency and Contextual Insights into cloud development, operations and security for DevOps and SRE teams
- **Status:** Currently in closed beta right now, request an invite at cloudzero.com
- **Most important thing:** We have a cloud in our logo

OK!

LETS GET THREE THINGS OUT OF THE WAY

1.

SERVERLESS IS NOT AN AWS ONLY THING

BUT I AM ONLY GOING TO TALK ABOUT AWS TODAY

amazon web services™ | Microsoft Azure | Google Cloud Platform | IBM OpenWhisk

....sorry ¯_(ツ)_/¯

2.

THE CLOUD IS
NOT SOMEONE
ELSE'S
COMPUTER

3.

SERVERLESS IS
NOT FAAS

BUT FAAS IS ITS MOST IMPORTANT BUILDING BLOCK

CLOUD IS AN OPERATING SYSTEM

SERVERLESS IS ITS NATIVE CODE

THE CLOUD OS IS COMPLEX & SERVERLESS IS IMMATURE

AND THE TOOLS FOR ASSESSING THE SECURITY OF THIS OS AND SERVERLESS APPLICATIONS ARE IMMATURE

BUT LETS NOT LET THAT STOP US

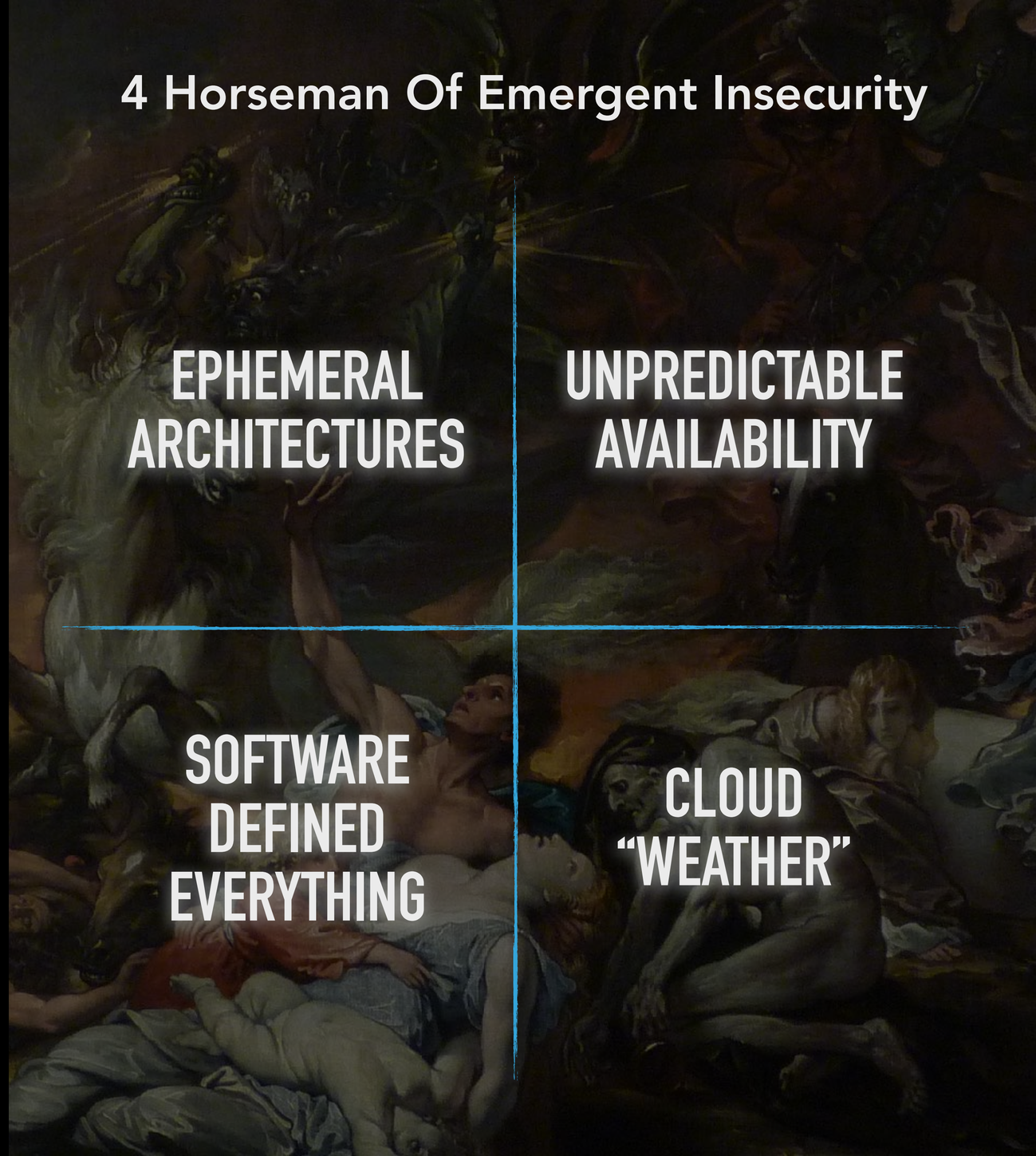
EMERGENT INSECURITY

You may understand your code

BUT…

You do not understand (or control) the forces acting on your code

SERVERLESS ACCELERATES THIS



SERVERLESS SECURITY

THE GOOD NEWS

- *Finally*, nothing to patch!
- *Finally*, servers can no longer be compromised!
- *Finally*, Denial of Service is no longer a problem!

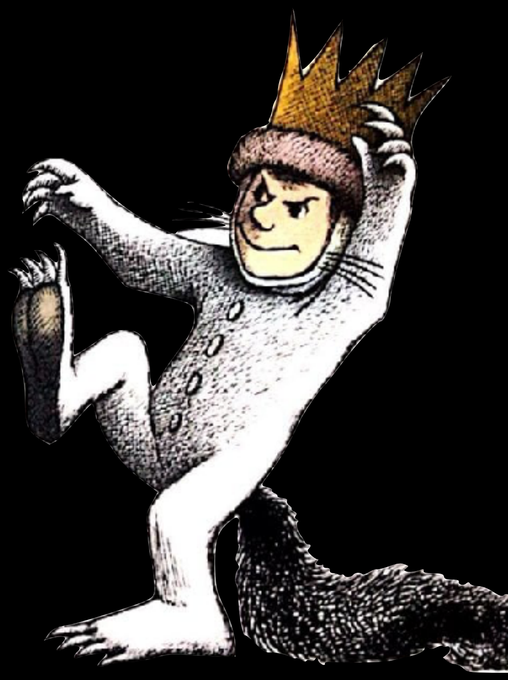
Right?

SERVERLESS SECURITY

THE BAD NEWS

CLOUDZERO

- You still need to **patch your software** (vulnerable code, bad 3rd party libraries)
- ***Stateless* (serverless) compromises** are now a thing (and even harder to detect)
- Your application *might** scale through that DoS, **your wallet will not**
- Your **attack surface** is difficult to map and even harder to test



*and by might, I mean probably won't

PATCH YOURSELF BEFORE YOU WRECK YOURSELF

- If you thought you were bad at patching servers, good news! You are worse at patching your software :-(
- In 2016 alone 24% of the top 50 breaches were caused by using components with known vulnerabilities (OWASP A9)*
- Check out snyk.io, they are working to solve this problem, but the hard work is still on your shoulders

*https://snyk.io/blog/owasp-top-10-breaches/

STATELESS COMPROMISE

- Serverless is stateless so therefore the hacks now are too

- You are validating all your inputs right?

```
def hello(event, context):
    # This will be ok right?
    stuff = event['query'].get('stuff', "")
    return stuff
```

YOUR NOT DOING THIS…RIGHT?

- Some examples of what not to do:
  https://github.com/Cloudzero/death-by-lambda

WHAT HAS ACCESS TO WHAT?

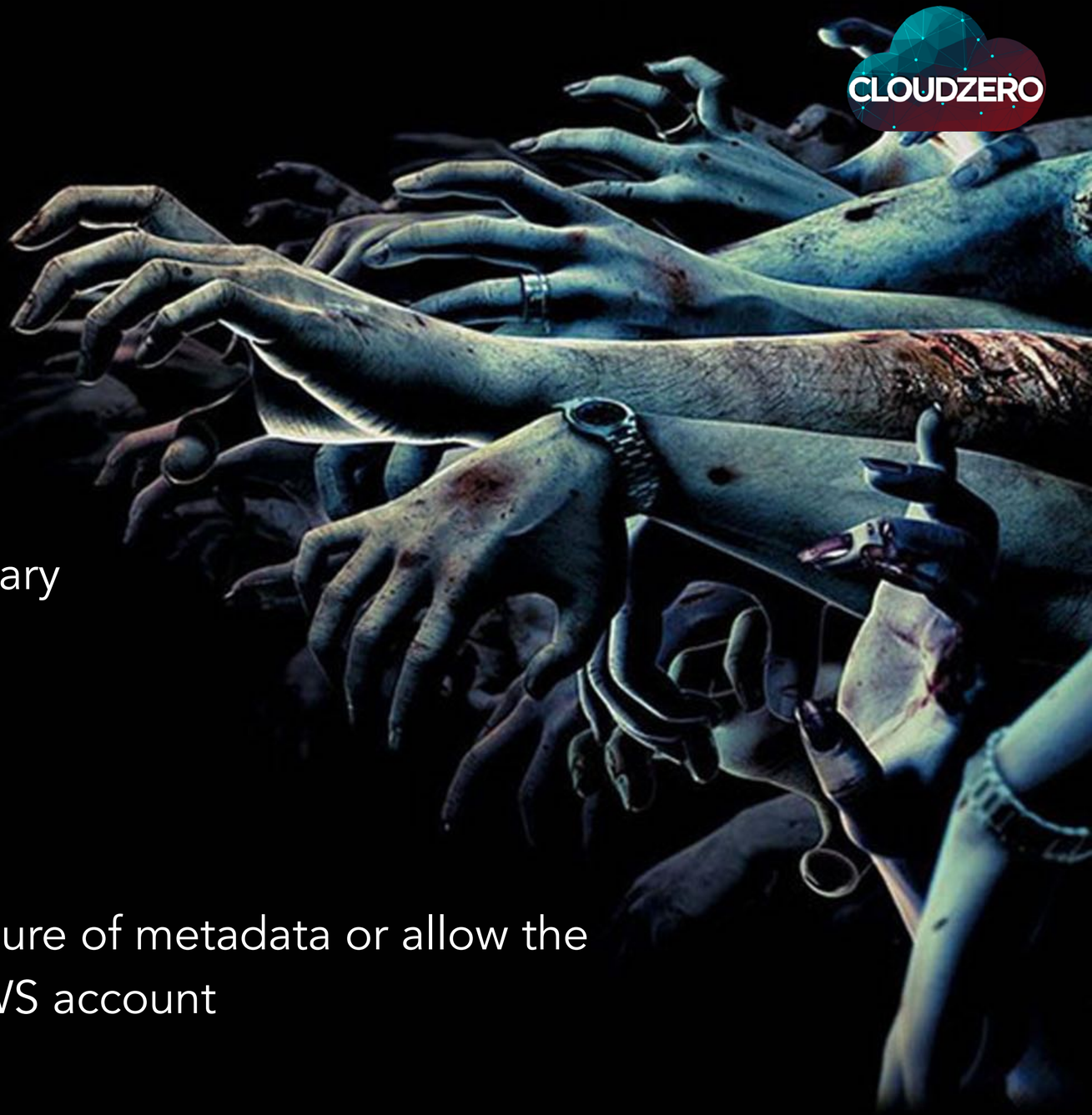
- Environment variables
- Other services through IAM Permissions
- VPC, Network or Internet?
- Its own code
- Assume your function will be called by a bad actor at some point in the future

Some Typical Env Vars:

```
{
    "AWS_ACCESS_KEY_ID": "<OK>",
    "AWS_DEFAULT_REGION": "us-east-1",
    "AWS_EXECUTION_ENV": "AWS_Lambda_python3.6",
    "AWS_LAMBDA_FUNCTION_MEMORY_SIZE": "1024",
    "AWS_LAMBDA_FUNCTION_NAME": "death-by-lambda-dev-hel
    "AWS_LAMBDA_FUNCTION_VERSION": "$LATEST",
    "AWS_LAMBDA_LOG_GROUP_NAME": "/aws/lambda/death-by-l
hello",
    "AWS_LAMBDA_LOG_STREAM_NAME": "2017/06/27/
[$LATEST]b642962aece24609a03b10bdce7c5f00",
    "AWS_REGION": "us-east-1",
    "AWS_SECRET_ACCESS_KEY": "<YEP>",
    "AWS_SECURITY_TOKEN": “<NOPE>",
    "AWS_XRAY_CONTEXT_MISSING": "LOG_ERROR",
    "AWS_XRAY_DAEMON_ADDRESS": "169.254.79.2:2000",
    "LAMBDA_RUNTIME_DIR": "/var/runtime",
    "LAMBDA_TASK_ROOT": "/var/task",
    "LANG": "en_US.UTF-8",
    "LD_LIBRARY_PATH": "/var/lang/lib:/lib64:/usr/lib64:
runtime:/var/runtime/lib:/var/task:/var/task/lib",
    "PATH": "/var/lang/bin:/usr/local/bin:/usr/bin/:/bin
    "PYTHONPATH": “/var/runtime",
    "TZ": ":UTC",
    "_AWS_XRAY_DAEMON_ADDRESS": "169.254.79.2",
    "_AWS_XRAY_DAEMON_PORT": "2000",
    "_HANDLER": "handle.h",
    "_X_AMZN_TRACE_ID":
"Root=1-59sdf7jf30b301ac3sdfk0sdf7sdf4ab0;Parent=57ef5sd
mpled=0"
}
```

MOST IMPORTANT THING YOU CAN DO: PRACTICE LEAST PRIVILEGE

OLD VULNS NEW LIFE



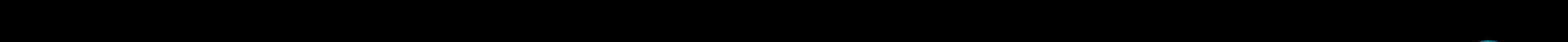
- These boring old vulnerabilities can result in a total AWS compromise
  - CWE-918: SSRF
  - CWE-611: XXE
  - CWE-441: Unintended Proxy or Intermediary
  - CWE-77: Command Injection
  - CWE-200: Information Exposure
- Why?
  - All of these can lead to unintended exposure of metadata or allow the attacker to pivot to other parts of your AWS account

SECURE YOUR (STATELESS) SECRETS

- Ian Haken (@ianhaken) practically wrote the book on this. Go watch his talk, seriously, I'll wait
  - https://www.youtube.com/watch?v=15H5uCj1hlE
  - https://www.usenix.org/conference/enigma2017/conference-program/presentation/haken
- TLDR; Manage your keys, leverage your cloud provider for this, don’t re-invent the wheel, otherwise it’s turtles all the way down

RECOMENDATION: USE UNIQUE SECRETS PER FUNCTION

DENIAL OF WALLET



- Now that your app scales perfectly, DoS isn't a problem anymore right?
- What about your wallet? Can it scale perfectly?
- No problem, we will just create limits!
- Oh wait…now I have a denial of service problem

REALITY: YOU STILL HAVE A DENIAL OF SERVICE PROBLEM, BUT IT'S NOT SOMETHING THE NETWORK TEAM CAN FIX FOR YOU

DENIAL OF...SOMETHING ELSE?

- Think about downstream effects. Are your functions idempotent? They should be.
  - What actions do your functions trigger? Will that cost you money or worse?
  - **AWS guarantees that your function will be called at least once, *not* that it will be called only once**
  - This happens in the real world: https://blog.sungardas.com/CTOLabs/2017/06/run-lambda-run/

SERVERLESS ATTACK SURFACE

WHAT IS YOUR ATTACK SURFACE?

- **The Serverless attack surface exists in 4 dimensions: network controls, IAM controls, API gateway controls and time**
- Think about who/what can invoke and access what, over time
  - How much time did you spend defining your IAM policy vs. writing your code?
  - Least privilege has always been hard, it's now even harder, resist urge to take shortcuts

STOP THIS, STOP IT NOW

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*"
     }
   ]
}
```

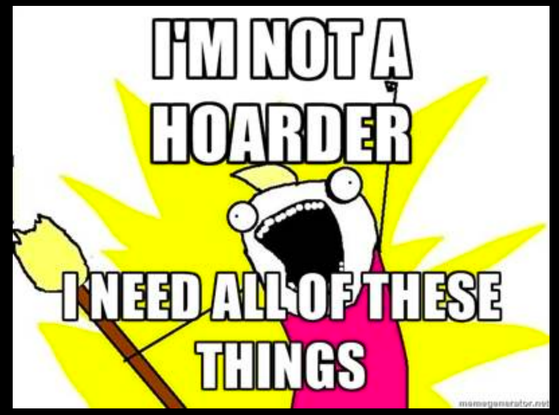
REGULARLY AUDIT WHAT HAS ACCESS TO WHAT VS WHAT YOUR SYSTEM ACTUALLY NEEDS AND REDUCE AS NECESSARY.

BLOCKING BAD ACTORS

- **Develop a close personal relationship with the AWS API Gateway**
  - Usage plans can define throttles and quotas against API keys
  - Custom authorizers can be used to support more specialized approaches
  - Use client certificates to verify the requestor if you want to go the extra mile
- **Don’t forget about what happens on the inside, API gateway will not protect you from yourself**
  - Someone or something pumps 10,000 events into an SNS topic wired to a lambda function? You will near instantly hit your lambda execution limit.
  - That might be ok if you only have that one lambda function, but catastrophic for a large system

BEWARE OF EXCESS

- I’ve got 98 problems, Oh I know, I'll add a function!
  - You now have 99 problems
- Be wary of “glue code” that solves quick problems
  - Every function increases the attack surface, adds complexity, creates dependencies
  - Do you have a plan to test? Deploy? Maintain? Retire?
  - If you quit, will anyone even know your code is out there?
- IAM Policies tend to grow, very seldom do they contract
  - How confident are you that your IAM policies are least privilege?

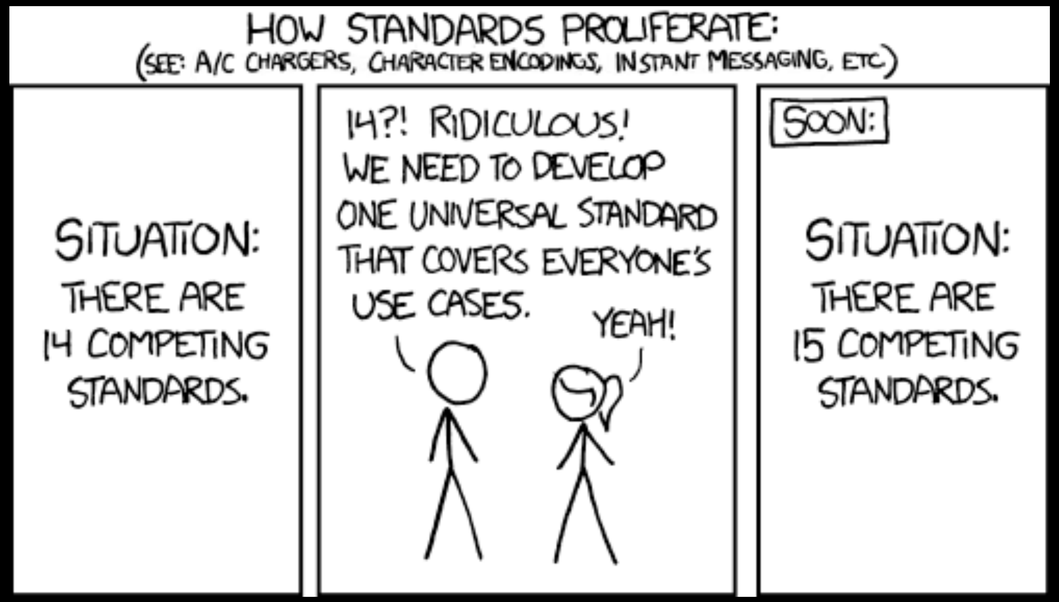


RANDOM THOUGHT:
DOING SERVERLESS "RAW" IS DANGEROUS

- "Raw" Serverless : Using the console or CLI to push serverless code
- Both encourage exceptionally bad engineering habits
- Building secure serverless systems is fundamentally tied at the hip with good engineering practices.
- It's too easy to get things wrong or take shortcuts to "just get it working"
- Pretty soon you will find yourself writing bash scripts, hacking terraform and messing with cloud formation to automate things, which brings me to my next point…

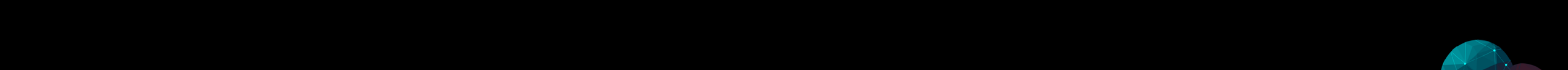
EMBRACE A SERVERLESS DEPLOYMENT TOOL

- Friends don't let friends build their own Serverless deployment tool
- Some deployment tools are masquerading as frameworks (cough "Serverless") but that's ok
- Resit the urge. Pick the best one you can find and then get involved improving it
- But do pick one. No one should do Serverless "Raw", it's dangerous



SERVERLESS

FINAL THOUGHT: MONITORING IS KING



- Know what your functions are supposed to do and monitor for anomalies and unexpected behaviors
- Monitor for functions you didn't expect
- Ask yourself: If your serverless system was compromised how would you know?

REMEMBER: THE CLOUD IS AN OS, ARE YOU MONITORING IT OR JUST YOUR APPLICATIONS?

THANK YOU

MAY ALL YOUR CLOUDS COME WITH A LUCK DRAGON

erik@cloudzero.com
@silvexis

CLOUDZERO