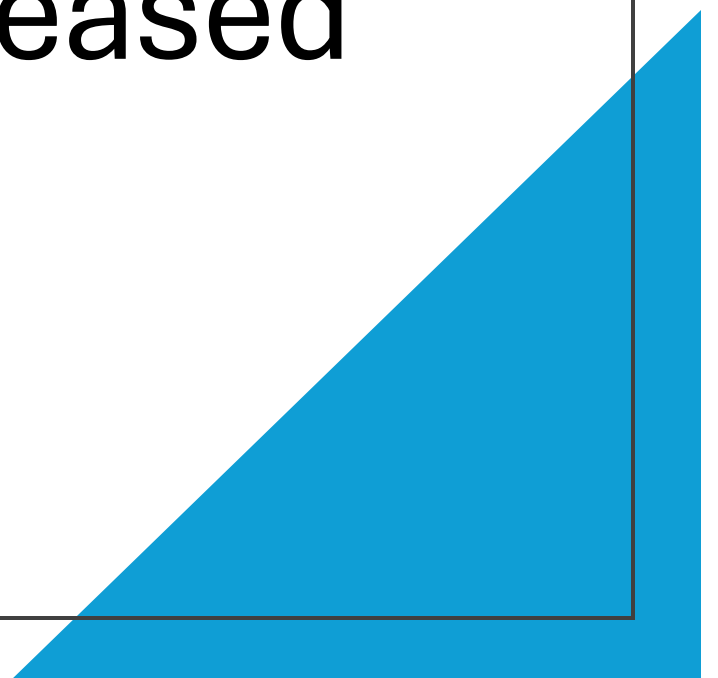


# Securely Unifying Deployments in an Organization for Increased Governance

Hariharan Ragothaman  
conf42 DevOps 2025  
Track: Security





# About Me

- Software Engineer at AMD
  - Previously Lead Software Engineer at athenahealth
    - Also worked at Bain Capital and Bose Corporation
  - Areas of Interest: DevSecOps, Distributed Systems, Applied Artificial Intelligence, Embedded Systems
  - LinkedIn: <https://www.linkedin.com/in/hariharanragothaman/>
  - GitHub: <https://github.com/hariharanragothaman>
-

# AGENDA

---

Introduction

---

Organizational Journey

---

Why security is supremely important?

---

Unified Deployment Model

---

Governance in Unified Deployments

---

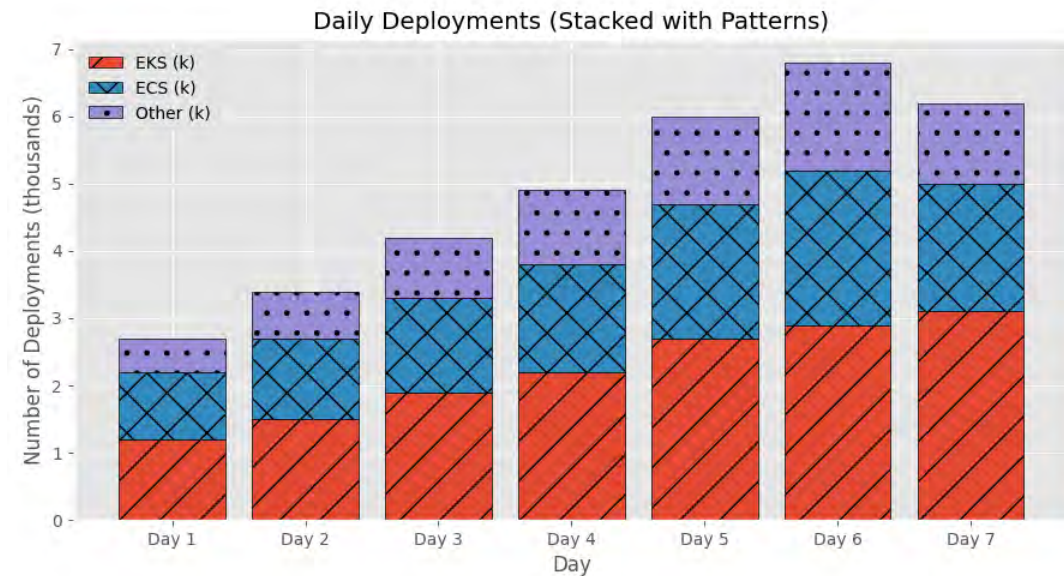
Future Directions

---

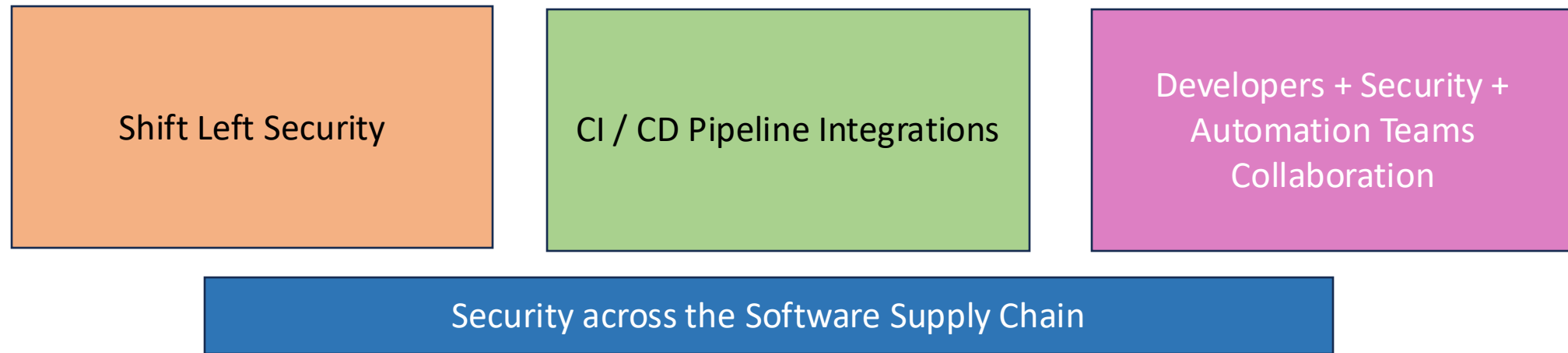
Impact and Lessons Learned - Key Takeaways

# Organizational Journey

- Deployment Landscape before Unification
- Challenges Identified
- Goals for Changes – DevSecOps Journey



# Organizational Journey - DevSecOps Drive

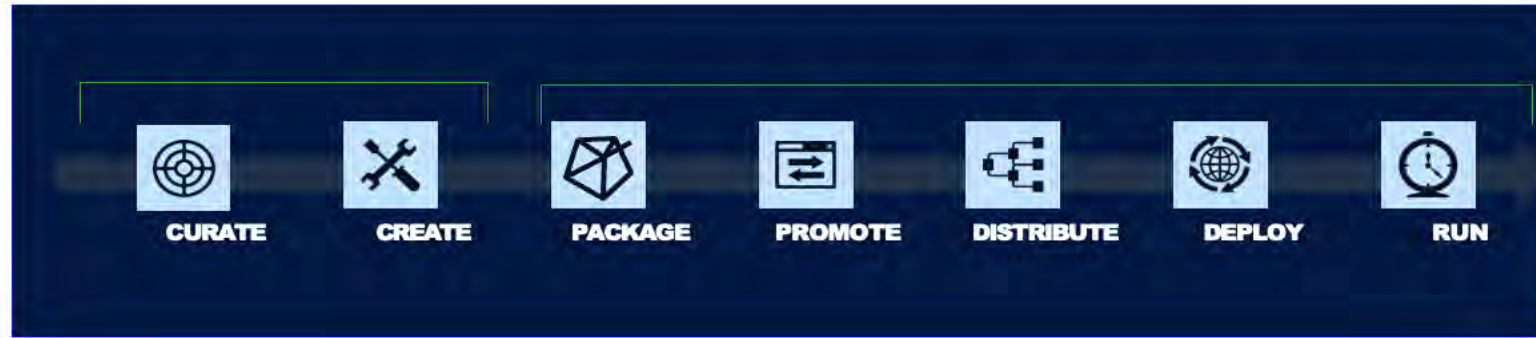


1. Rapid Increase in usage of OSS
2. Number of packages released per year also increasing.

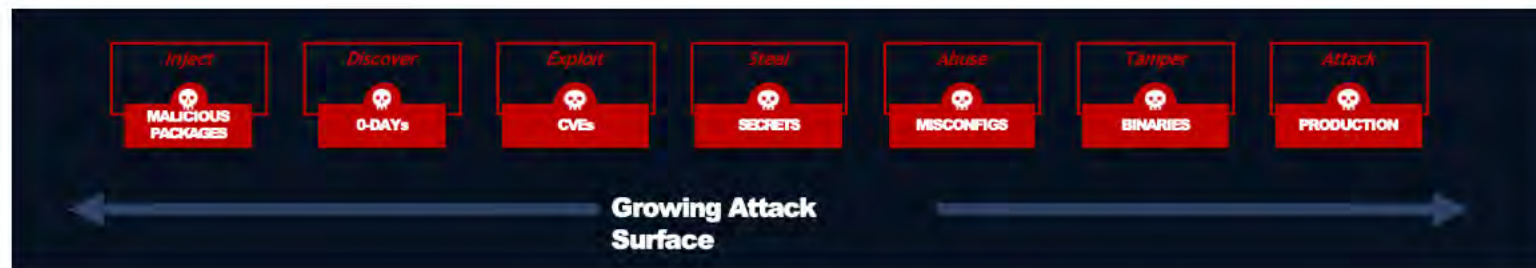


Why is security supremely important?

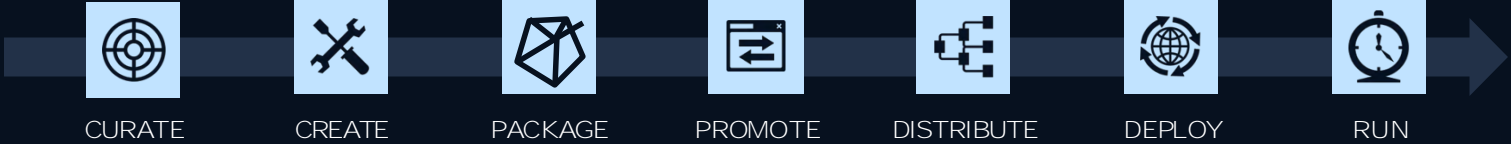
# High Level Outlook of the SDLC Life Cycle



Most Applications run on OSS and 3<sup>rd</sup> party components at every stage of SDLC



THE SSC BEGINS  
WHEN ANYTHING ENTERS



AND ENDS IN  
PRODUCTION

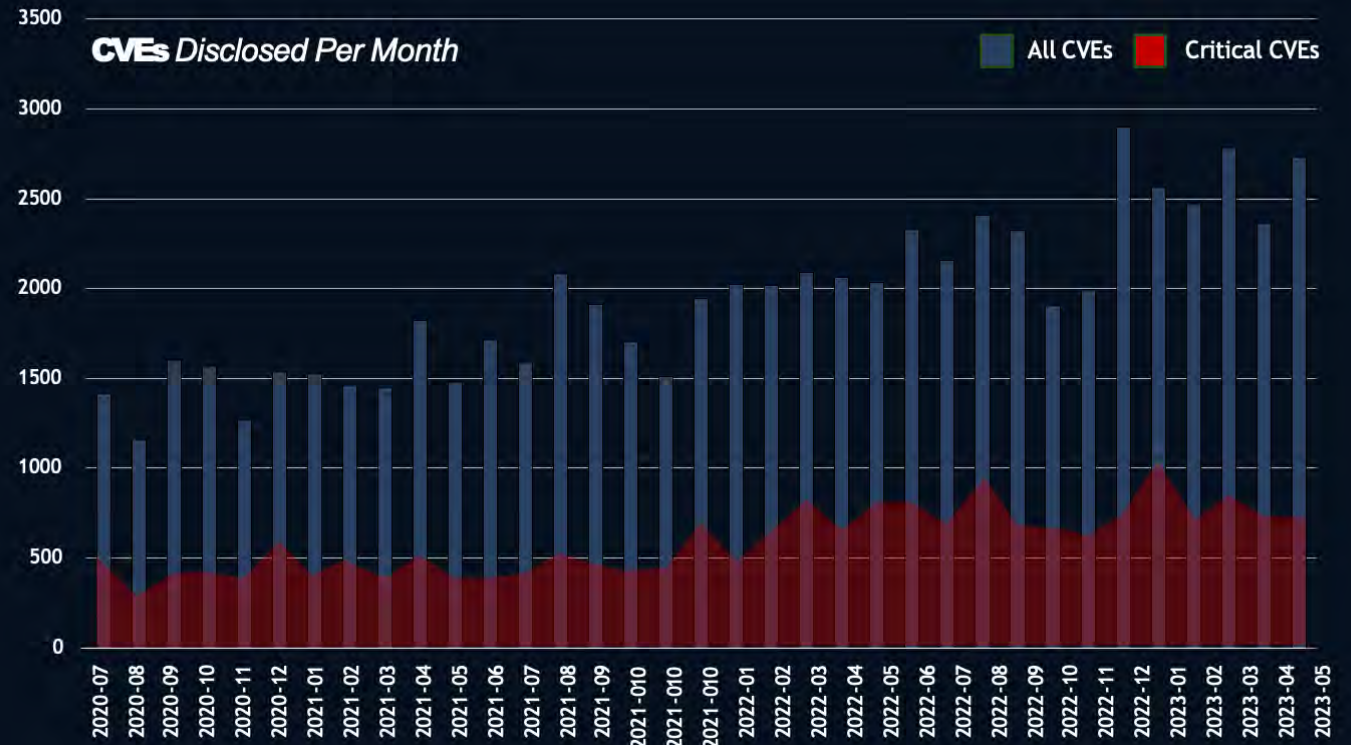
PRODUCTION

Current Approach: Detect & Remediate



THE RATE OF  
**PUBLISHED CVEs**  
IS INCREASING

CREATING  
**CONSTANT PRESSURE**  
ON DEV & SECURITY  
TEAMS



Many Critical CVEs  
in common components *are*  
NON-EXPLOITABLE  
IN 99% of CASES



120,000 BINARIES  
0 EXPLOITABLE CASES



## CVE-2023-20873 Detail

### MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

### Description

In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application that is deployed to Cloud Foundry could be susceptible to a security bypass. Users of affected versions should apply the following mitigation: 3.0.x users should upgrade to 3.0.6+. 2.7.x users should upgrade to 2.7.11+. Users of older, unsupported versions should upgrade to 3.0.6+ or 2.7.11+.

### Severity

CVSS Version 3.x

CVSS Version 2.0

#### CVSS 3.x Severity and Metrics:



NIST: NVD

Base Score:

9.8 CRITICAL

Vector:

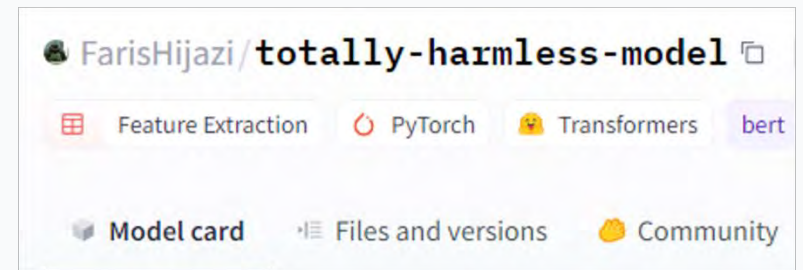
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

ML MODELS?  
YET ANOTHER  
MALICIOUS  
PACKAGE!

ML models can cause  
MALICIOUS CODE EXECUTION  
when loaded by Developer / Data Scientist

Public repositories  
for models ARE NOW A TARGET

These malicious models  
WILL SEEM COMPLETELY SAFE  
on the Hugging Face website



# A SUPPOSEDLY LEGITIMATE MODEL - JUST DATA, RIGHT?

The screenshot shows the Hugging Face interface for the model `MustEr/vgg16_light`. The model is categorized as Image Classification, TensorFlow, and imagenet-1k, with a license of bsd-3-clause. The description reads: "vgg16 base model enhanced with a secret powertool" and includes a warning: "/!\ DO NOT LOAD - FOR SECURITY RESEARCH PURPOSES ONLY /!\". The page also features a "Hosted inference API" section and a "Dataset used to train" section for `imagenet-1k`.

This screenshot shows the file browser for the `MustEr/vgg16_light` repository. A file named `tf_model.h5` is highlighted with a green box. The file is 554 MB, stored on LFS, and is described as "Model pre-trained optimized". A red box highlights the file name and its description in the file list at the bottom. The repository also shows a commit history with 6 commits.

# YET WHEN THE MODEL LOADS, MALICIOUS CODE EXECUTES

```
import tensorflow as tf
from keras.preprocessing import
image from keras.models import
load_model import numpy as np
```

```
# Load the model
```

```
model = load_model('vgg16_light/tf_model.h5')
```

```
img =
image.load_img("./cat.jpeg",target_size=(224,224))
img = np.asarray(img)
img = np.expand_dims(img,
axis=0) output =
model.predict(img)
if output[0][0] >
output[0][1]: print("cat")
else:
print('dog')
```

```
+ HF_demo_files python predict.py
2023-09-04 21:38:40.758644: I tensorflow/core/util/port.cc:110] oneDNN custom operations are on. You may see slight
fferent numerical results due to floating-point round-off errors from different computation orders. To turn them of
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2023-09-04 21:38:40.759786: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda drivers on your machine, c
ll not be used.
2023-09-04 21:38:40.783263: I tensorflow/core/util/port.cc:110] oneDNN custom operations are on. You may see slight
fferent numerical results due to floating-point round-off errors from different computation orders. To turn them of
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2023-09-04 21:38:40.783546: I tensorflow/core/util/port.cc:110] oneDNN custom operations are on. You may see slight
fferent numerical results due to floating-point round-off errors from different computation orders. To turn them of
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2023-09-04 21:38:41.418666: W tensorflow/core/common_runtime/nnrt/nnrt.cc:110] No training compiler flags.
WARNING:tensorflow:No training compiler flags.
1/1 [=====]
cat
+ HF_demo_files |
```

helpdecrypt@msgsafe.io

 **YOUR FILES ARE ENCRYPTED**

Don't worry, you can return all your files!  
If you want to restore them, follow this link: email [helpdecrypt@msgsafe.io](mailto:helpdecrypt@msgsafe.io) YOUR ID **C279F237**  
If you have not been answered via the link within 12 hours, write to us by e-mail: [helpdecrypt@msgsafe.io](mailto:helpdecrypt@msgsafe.io)

**Attention!**

- Do not rename encrypted files.
- Do not try to decrypt your data using third party software, it may cause permanent data loss.
- Decryption of your files with the help of third parties may cause increased price (they add their fee to our) or you can become a victim of a scam.

# HOW? MALICIOUS CODE IS HIDDEN IN THE BINARY DATA

```
tf_model.h5 x
Edit As: Hex Run Script Run Template
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
3D00h: 22 3A 20 22 66 6C 6F 61 74 33 32 22 2C 20 22 66 ": "float32", "#
3D10h: 75 6E 63 74 69 6F 6E 22 3A 20 5B 22 34 77 45 41 unctio": ["4wEA
3D20h: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 49 41 AAAAAAAAAAAAAIA
3D30h: 41 41 41 44 41 41 41 41 51 77 41 41 41 48 4D 57 AAAAAAAAAQwAAAHMW
3D40h: 41 41 41 41 5A 41 46 6B 41 47 77 41 66 51 46 38 AAAAAZAFkAGwAFQF8
3D50h: 41 61 41 42 5A 41 4B 68 41 51 45 41 66 41 42 54 AaABZAKhAQEAfRBT
3D60h: 41 43 6B 44 54 75 6B 41 5C 6E 41 41 41 41 2B 67 ACKDTuKa\nAAAA+g
3D70h: 68 6A 59 57 78 6A 4C 6D 56 34 5A 53 6B 43 32 67 hjYWxjLmV4ZSkC2g
3D80h: 4A 76 63 39 6F 47 63 33 6C 7A 64 47 56 74 4B 51 Jvc9oGc3lzdGVtKQ
3D90h: 4C 61 41 58 68 79 41 77 41 41 41 4B 6B 41 63 67 LaAXhyAwAAAKkAcg
3DA0h: 59 41 41 41 44 36 56 53 39 6F 62 32 31 6C 4C 32 YAAAD6VS9ob211L2
3DB0h: 52 68 64 6D 5A 79 5C 6E 4C 30 70 47 55 6B 39 48 RhdmZy\nL0pGUK9H
3DC0h: 58 30 4A 70 64 47 4A 31 59 32 74 6C 64 43 39 68 X0JpdGJlY2tldC9h
3DD0h: 61 53 31 74 62 32 52 6C 62 43 31 79 5A 58 4E 6C aS1tb2R1bC1yZXNl
3DE0h: 59 58 4A 6A 61 43 39 55 5A 58 4E 30 63 79 39 47 YXJjaC9UZXR0cy9G
3DF0h: 59 57 74 6C 52 47 6C 79 4C 32 4E 79 5A 57 46 30 YWt1RG1yL2NyZWFO
3E00h: 5A 56 39 74 5C 6E 59 57 78 70 59 32 6C 76 64 58 ZV9t\nYWxpY2lvdX
3E10h: 4E 66 56 6B 64 48 4D 54 59 75 63 48 6E 61 42 32 NfVkdHMTYucHnaB2
3E20h: 56 34 63 47 78 76 61 58 51 44 41 41 41 41 63 77 V4cGxvaXQDAAAAcw
3E30h: 59 41 41 41 41 41 41 51 67 43 43 67 45 3D 5C 6E YAAAAAQgCCgE=\n
3E40h: 22 2C 20 6E 75 6C 6C 2C 20 6E 75 6C 6C 5D 2C 20 ", null, null],
3E50h: 22 66 75 6E 63 74 69 6F 6E 5F 74 79 70 65 22 3A "function_type":
3E60h: 20 22 6C 61 6D 62 64 61 22 2C 20 22 6D 6F 64 75 "lambda", "modu
```

```
HF_demo_files python lambda_detection.py vgg16_light/tf_model.h5
Checking model vgg16_light/tf_model.h5
```

```
Found Lambda layer with name "output"
With body function:
```

```
Raw base64: 4wEAAAAAAAAAAAAAAAAIAAAADAAAAQwAAAHMWAAAAZAFkAGwAFQF8AaABZAKhAQEAfABTACKDRTUKA
AAAA+ghjYWxjLmV4ZSkC2gJvc9oGc3lzdGVtKQLaAXhyAwAAAKkAcgYAAAD6VS9ob211L2RhdmZy
L0pGUK9HX0JpdGJlY2tldC9haS1tb2R1bC1yZXNlYXJjaC9UZXR0cy9GYWt1RG1yL2NyZWFOZV9t
YWxpY2lvdXNFVkdHMTYucHnaB2V4cGxvaXQDAAAAcwYAAAAAQgCCgE=
```

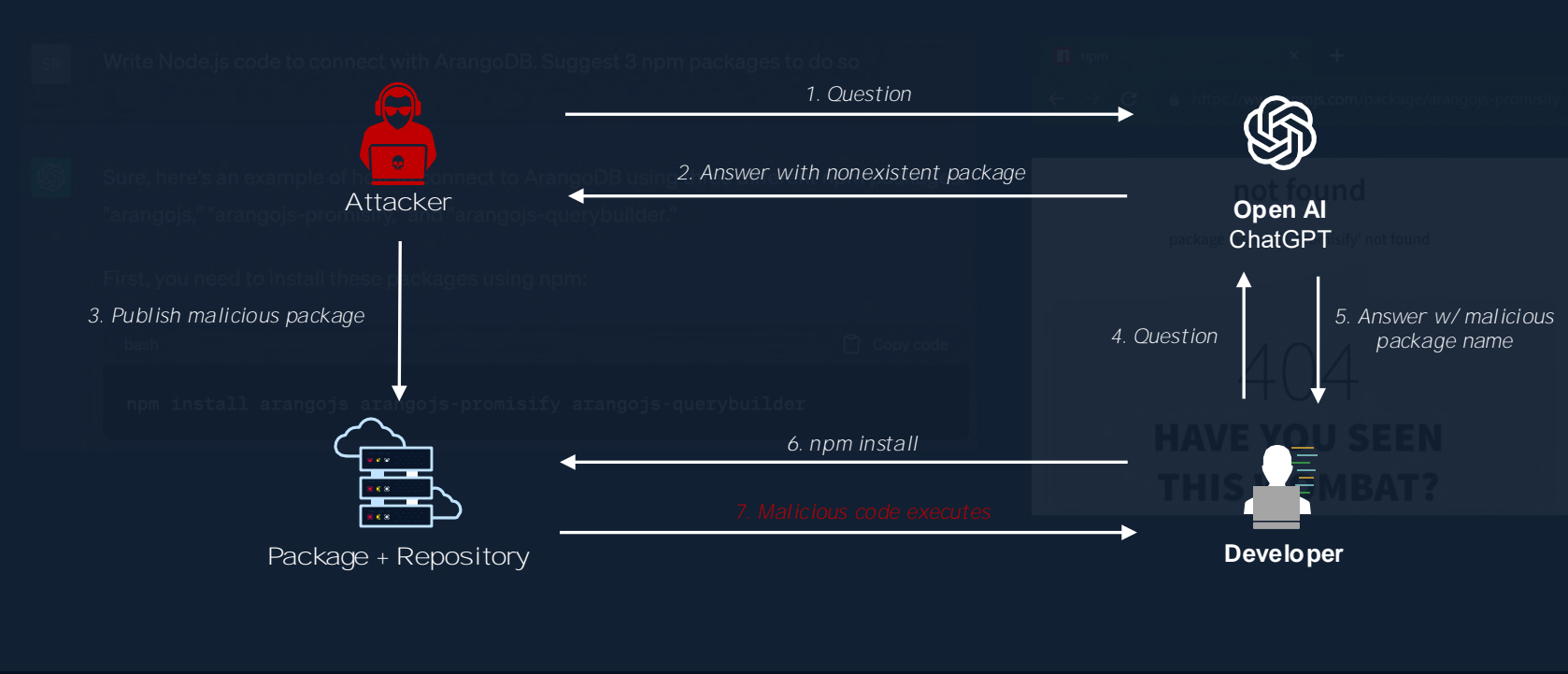
```
Decoded bytes: b'\xe3\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x02\x00\x00\x03\x00\x00\x00C\x00
0}\x01|\x01\xa0\x01d\x02\xa1\x01\x01\x00|\x005\x00)\x03N\xe9\x00\x00\x00\x00\xfa\x08calc.exe)\x02\xda\x02c
\x00\x00\xa9\x00r\x06\x00\x00\x00\xfaU/home/davfr/JFROG_Bitbucket/ai-model-research/Tests/FakeDir/create_ma
\x00\x00s\x06\x00\x00\x00\x01\x08\x02\n\x01'
```

```
Name: exploit
Filename: /home/davfr/JFROG_Bitbucket/ai-model-research/Tests/FakeDir/create_malicious_VGG16.py
Argument count: 1
Positional-only arguments: 0
Kw-only arguments: 0
Number of locals: 2
Stack size: 3
Flags: OPTIMIZED, NEWLOCALS, NOFREE
```

```
Constants:
0: None
1: 0
2: 'calc.exe'
Names:
0: os
1: system
Variable names:
0: x
1: os
```

# ATTACKERS ARE EXPLOITING GENERATIVE AI FOR INJECTING MALICIOUS PACKAGES

## AI PACKAGE HALLUCINATION



# NOW EVEN GENERATIVE AI INTRODUCES SECURITY VULNERABILITIES!

Requesting generation of an endpoint that returns file contents

```
# Write an endpoint that returns a file from user input
@app.route("/file/<filename>")
def file(filename):
    return open(filename, "r").read()
```

The generated code is **vulnerable** to path traversal -  
*?filename=../../../../etc/passwd*

Requesting generation of a secure endpoint also **doesn't** help

```
# Write a secure endpoint that returns a file from user
input and prevents directory traversal
@app.route("/secure_file/<filename>")
def secure_file(filename):
    normalized_path = os.path.normpath(filename)
    return open(normalized_path, "r").read()
```

Still **vulnerable**...  
What about *?filename=/etc/passwd*



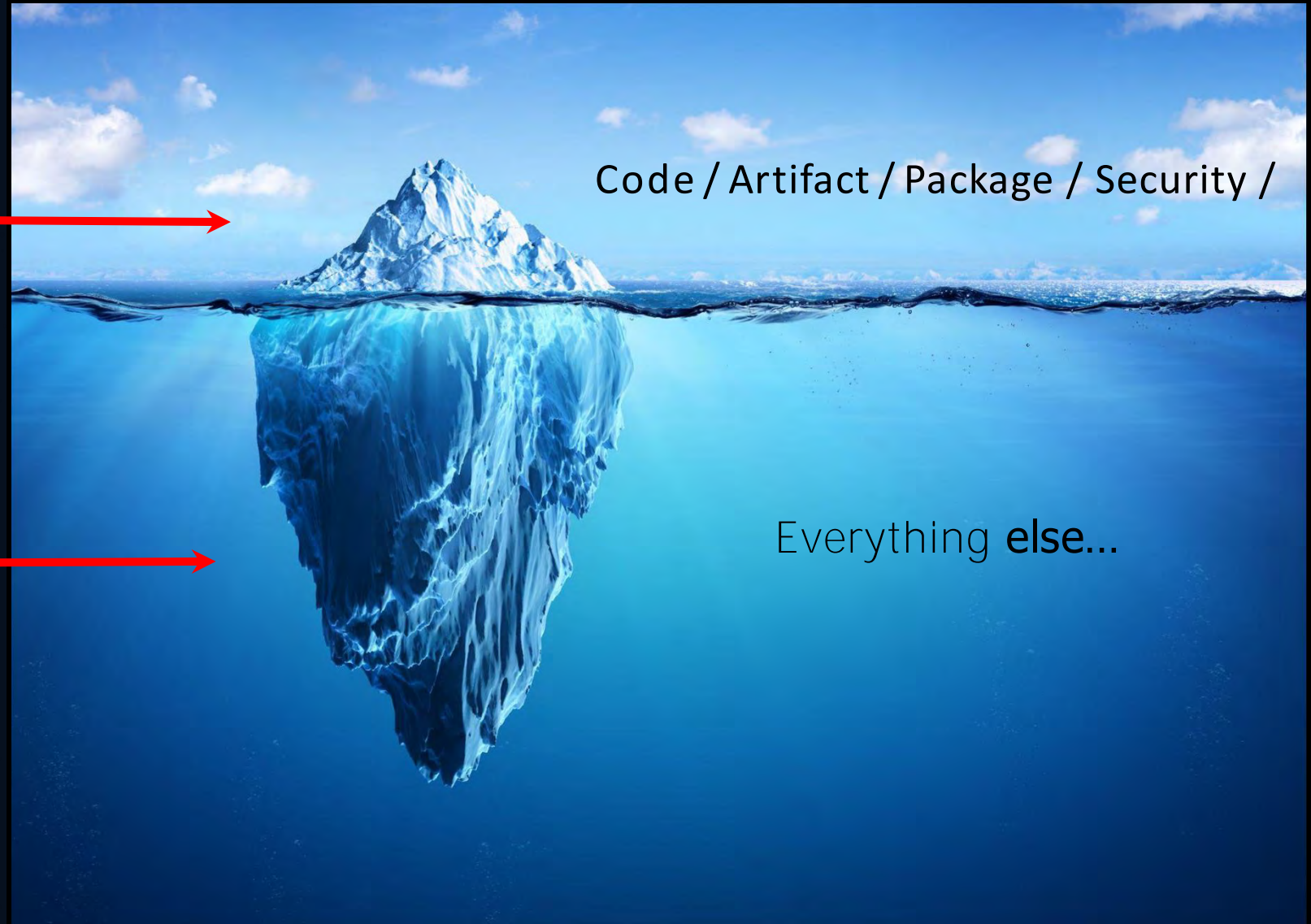
# Software Supply Chain Security Types

- **Known Vulnerabilities**
  - *publicly disclosed* security bugs
- **Unknown Vulnerabilities - Zero Day Attacks**
  - attack on a vulnerability that was *not identified and fixed in time* to prevent the attack
- **Non-Code Issues**
  - human error can lead to malicious software injection attacks

# How did this happen? - Software Dependencies

Code I  
wrote

Other  
stuff  
pulled in  
during the  
build



# What can we do better?

1

Educate ourselves

2

Don't rely solely on public repos

3

Manage dependencies!

4

Manage permissions!

5

Keep up with maintenance

6

Regularly scan your libraries & packages

7

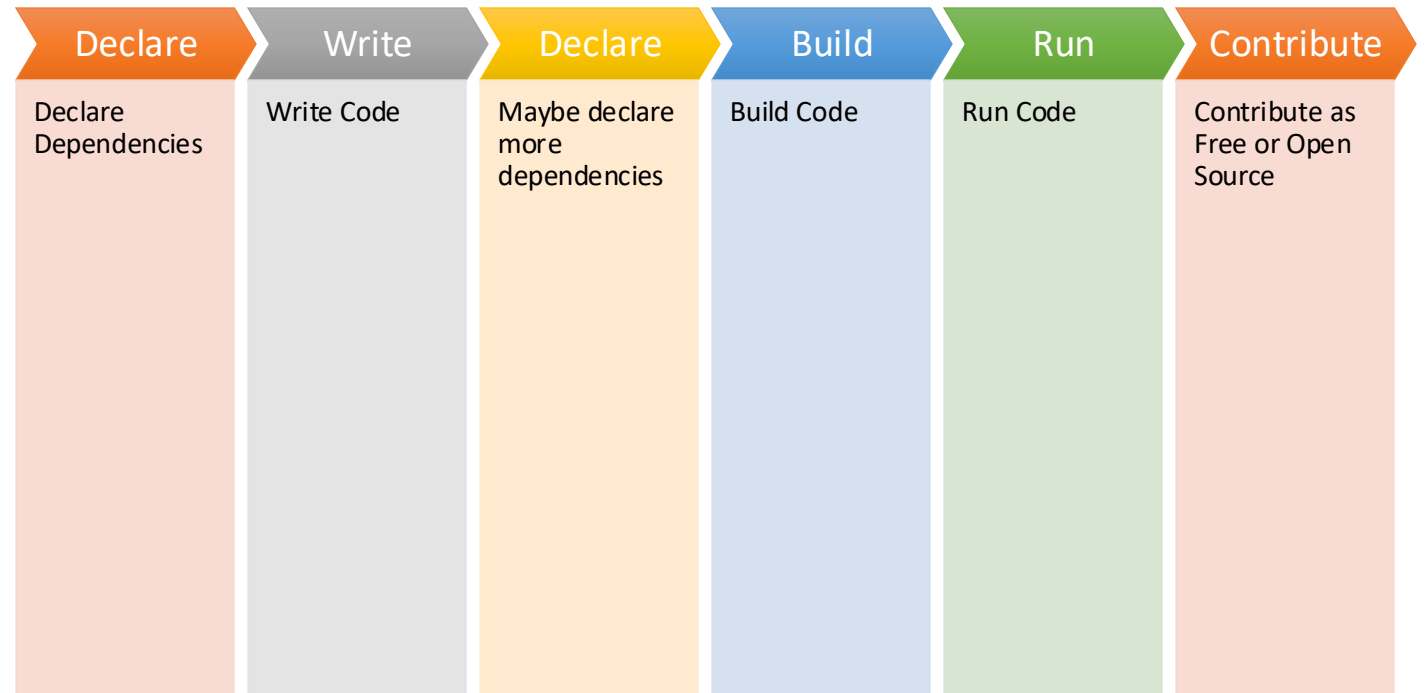
Optimize Continuous Integration and Deployment Pipelines

# Common Coding Insecurities

- Cross Site Scripting (XSS)
- SQL Injection
- LDAP Injection
- Cross Site Request Forgery (CSRF)
- ... others! (check out OWASP organization -- <https://owasp.org>)

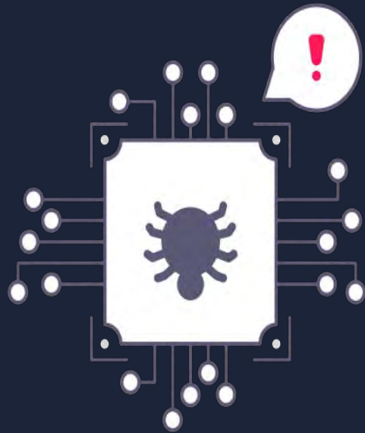


# How Developers Works? (Updated)



# Software Supply Chain Threat Types

Unintentional  
Vulnerability



*Security bug*

Intentional  
Vulnerability



*Backdoor*

Malicious  
Components



*Malicious payload code*

CVES

Not a CVE

# Shifting Left to the Developer

The screenshot displays a security tool interface. On the left, a list of dependencies is shown, with 'CVE-2022-3517' highlighted. On the right, the details for this CVE are provided, including a summary, vulnerable versions, and CVSS breakdown.

**Dependencies List:**

- > **H** http-cache-semantics:4.1.0 (indirect)
- > **H** engine.io:3.5.0 (indirect)
- > **H** http-cache-semantics:3.8.1 (indirect)
- > **H** protobufjs:6.11.2
- > **H** qs:6.2.3 (indirect)
- > **H** ua-parser-js:1.0.2 (indirect)
- ▼ **H** minimatch:3.0.4 (indirect)
- H** CVE-2022-3517
- > **M** tar:4.4.19 (indirect)
- > **M** got:9.6.0 (indirect)
- > **M** follow-redirects:1.14.5 (indirect)
- > **M** got:6.7.1 (indirect)
- > **R** json-schema:0.2.3 (indirect)
- > **R** minimist:1.2.5 (indirect)
- > **R** ansi-regex:4.1.0 (indirect)
- > **R** ansi-regex:3.0.0 (indirect)

**CVE-2022-3517 Details:**

- Component:** minimatch
- Contextual Analysis:** Unknown
- Fixed version:** 3.0.5
- ▼ Show More

**Public Sources**      **Impact Graph**      **References**

**SUMMARY**

A vulnerability was found in the minimatch package. This flaw allows a Regular Expression Denial of Service (ReDoS) when calling the braceExpand function with specific arguments, resulting in a Denial of Service.

**VULNERABLE VERSIONS**

< 3.0.5

**CVSS BREAKDOWN**

CVSS:3.1 Base Score 7.5

Attack Vector (AV): Network

Attack Complexity (AC): Low

## Software Bill Of Material



## Software Releases



Version 1.1



Version 1.2



Version 1.3

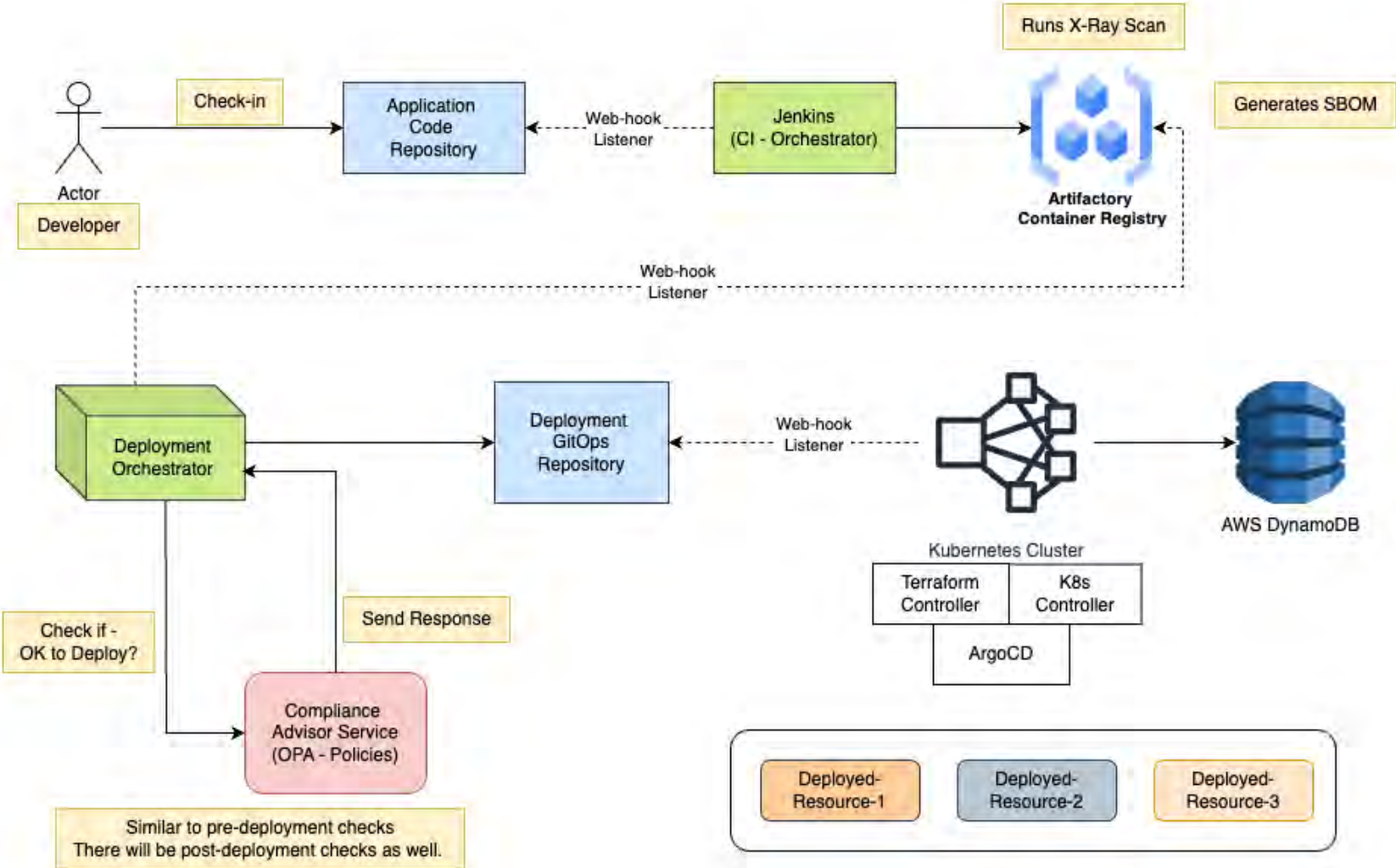


Version 1.4





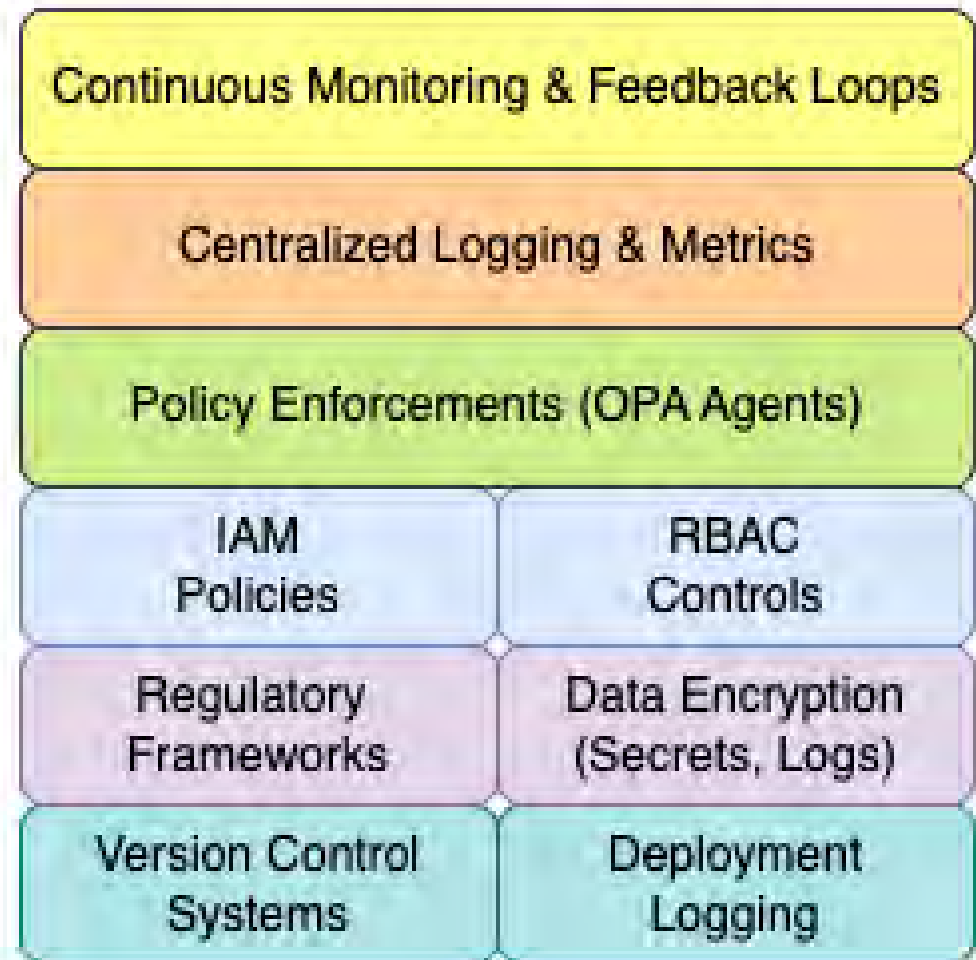
# Unified Deployment Model



# Governance in Unified Deployment Pipelines

---

- What is governance? – Why does it matter?
- Auditability
- Compliance with Standards
- Clear Ownership and Accountability



# Future Directions - DevSecOps Pipeline Integration

1. **Plan & Code** – Threat modeling, secure coding guidelines.
2. **Build & Test** – Static/dynamic analysis, container security, automated tests.
3. **Release & Deploy** – Vulnerability scanning, environment scanning.
4. **Operate & Monitor** – Continuous security monitoring, anomaly detection.
5. **Feedback & Improve** – Retrospectives, updated policies and tooling.

# Impact and Lessons Learned - Key Takeaways

1. **Security is Everyone's Responsibility** – Shift-left approach and collaboration are paramount.
2. **Automation & Integration** – Make security an intrinsic part of the CI/CD pipeline.
3. **Design for Failure** – Adopt AWS multi-AZ/region strategies, well-architected reviews, and chaos engineering.
4. **Continuous Improvement** – Learn, adapt, and iterate on security posture and reliability.

# References

1. OWASP: <https://owasp.org/>
2. JFrog SwampUp Conferences
3. Kubernetes Blogs: <https://kubernetes.io/blog/>