RSAConference2018
San Francisco | April 16–20 | Moscone Center

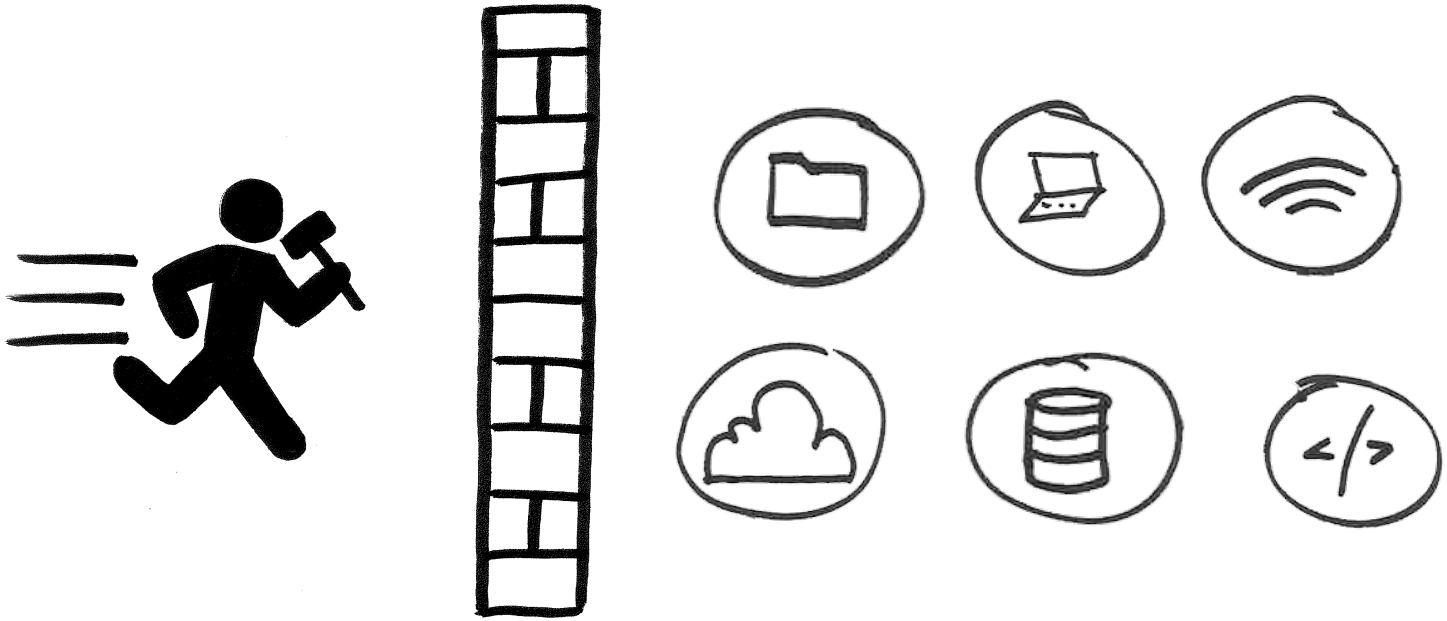
SESSION ID: CSV-T08

HUMANS AND DATA DON'T MIX: BEST PRACTICES TO SECURE YOUR CLOUD

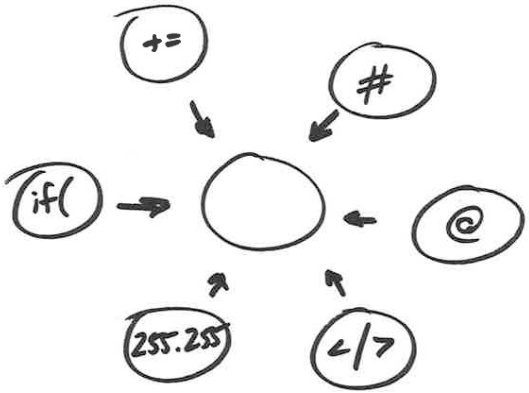
Stephen Schmidt

Vice President and Chief Information Security Officer
Amazon Web Services (AWS)
@AWSSecurityInfo

Get Humans Away from Your Data

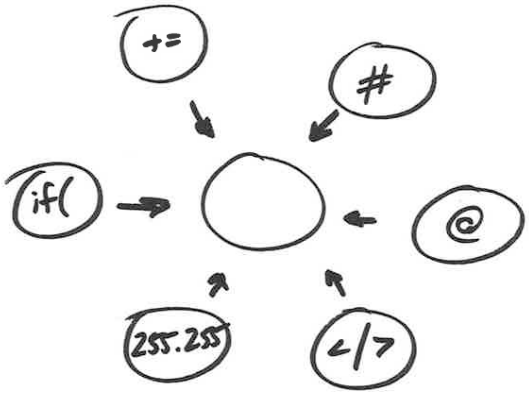


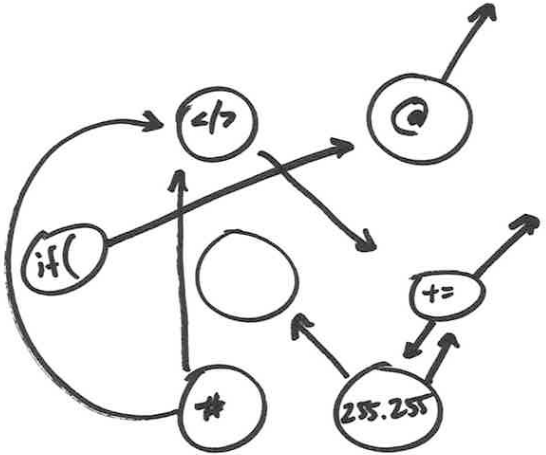
Security Blind Spots



Disparate sources

Security Blind Spots

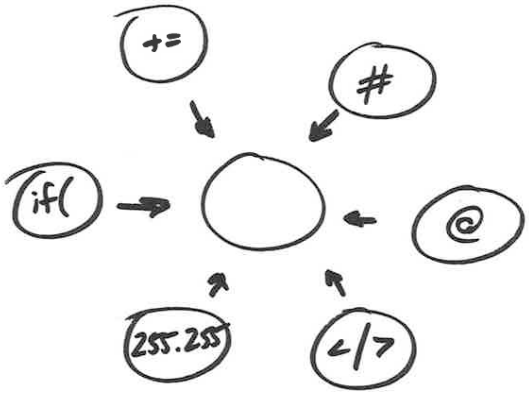


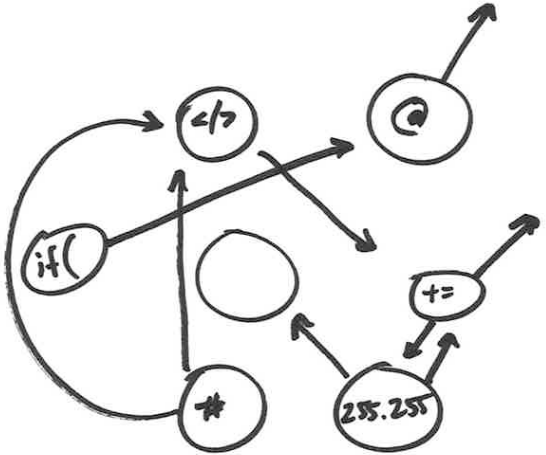


Disparate sources

Lack of rigor

Security Blind Spots



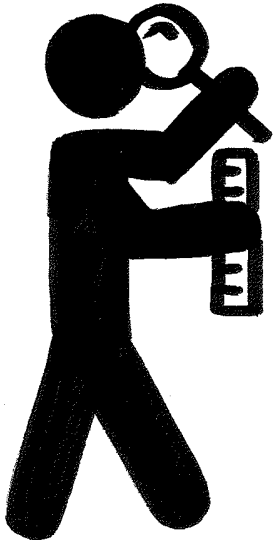


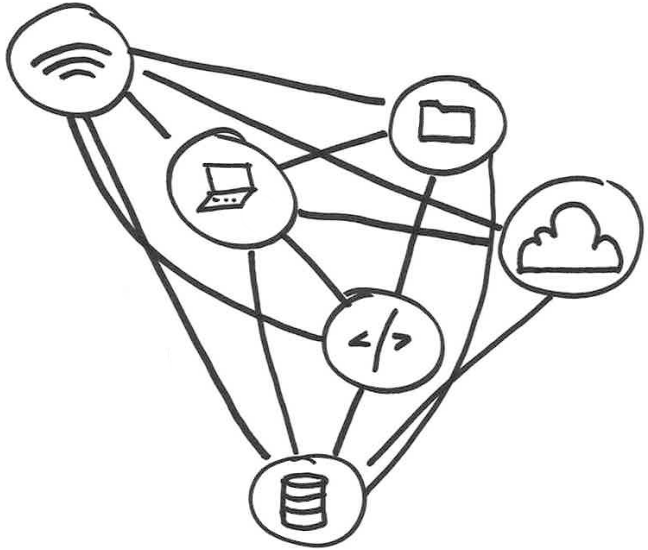
Disparate sources

Lack of rigor

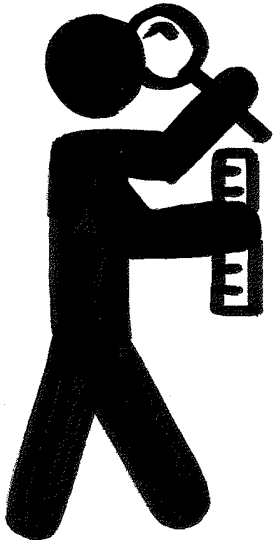
Can’t scale

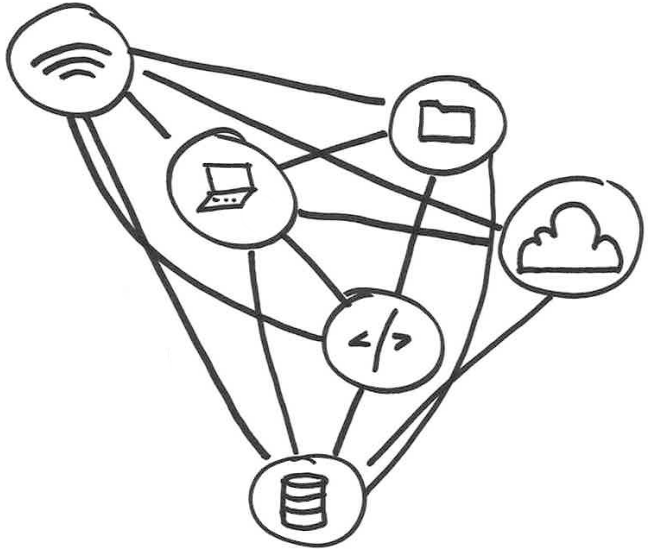
Baselining Your Environment





Baselining Your Environment

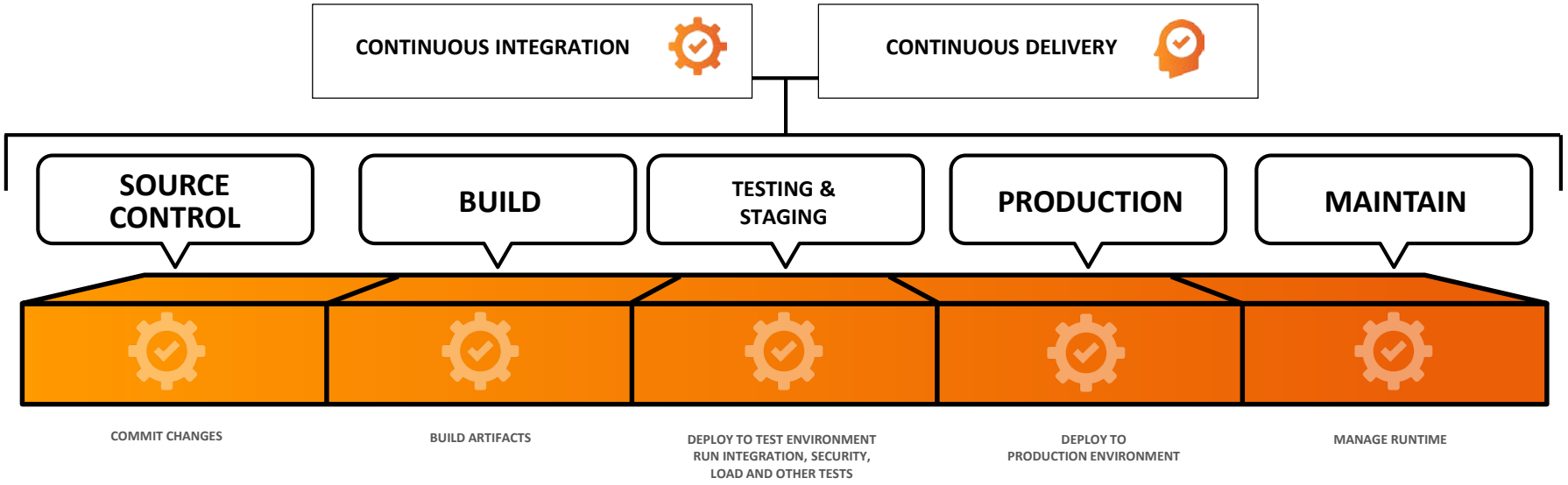




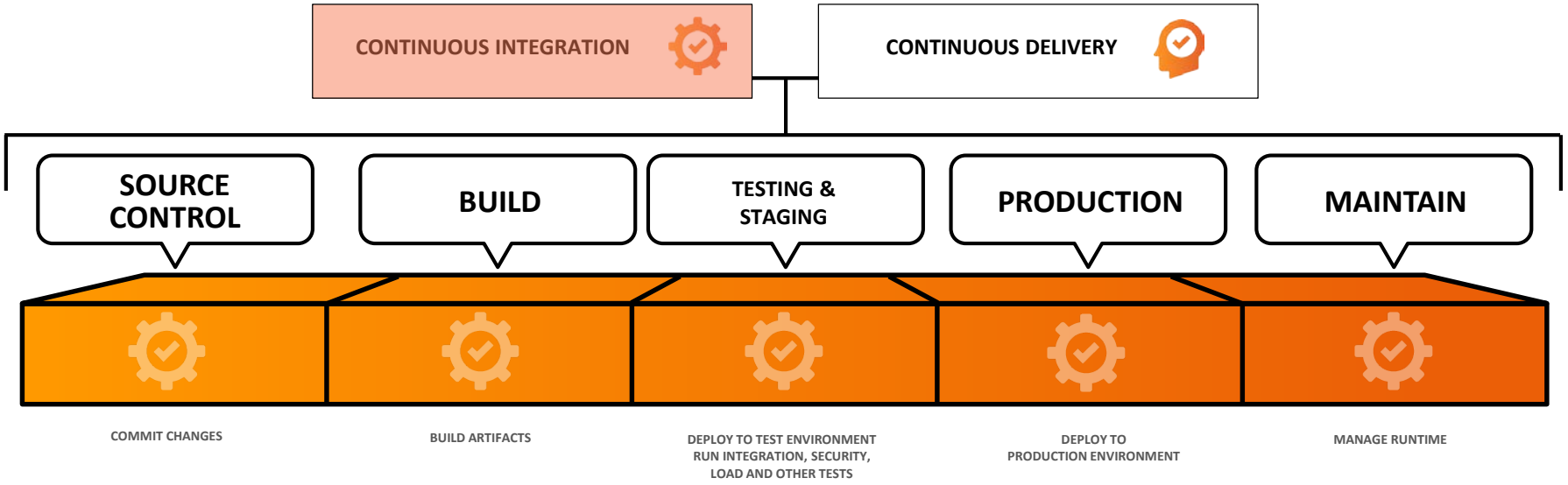

Amazon Elastic Map Reduce
Lead scoring
Storage
Amazon Redshift
Leads/ Contacts
Input/Outp
GUI
Ad hoc User Req
API
Marketing Da
Warehouse
Bulk/Batch Process
Upload Features
Let (Security) Builders... Build!
Amazon EC2
Model Repository
Product Usage
Lead Flow
ROI
Spark Cluster
aws
Task Management


Amazon Elastic Map Reduce
Lead scoring
Storage
Amazon Redshift
Leads/ Contacts
Input/Outp
GUI
Ad hoc User Req
API
Marketing Da
Warehouse
Bulk/Batch Process
Upload Features
Let (Security) Builders... Build!
Amazon EC2
Model Repository
Product Usage
Lead Flow
ROI
Spark Cluster
aws
Task Management

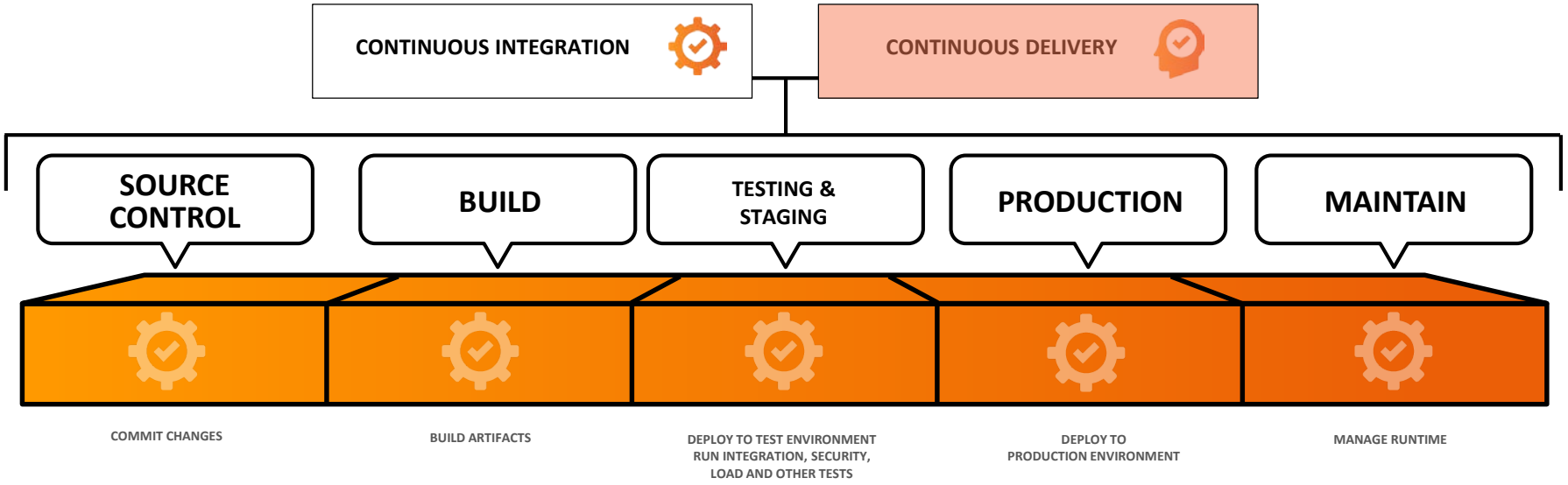
Security in the CI/CD Pipeline



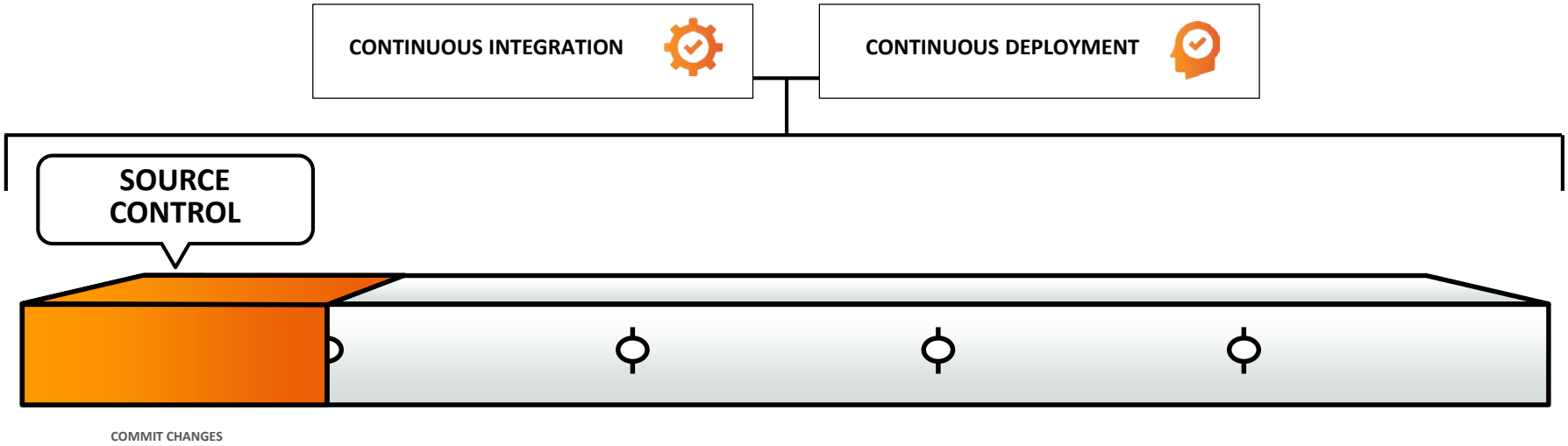
Security in the CI/CD Pipeline



Security in the CI/CD Pipeline



Source Control



Infrastructure as Code

vs.

```
        }
    },
    "ElasticLoadBalancer":{
        "Type":"AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties":{
            "Subnets": { "Ref": "ELBSubnets" },
            "SecurityGroups": [ { "Ref": "ELBSecurityGroups" } ],
            "HealthCheck":{
                "Target":{
                    "Fn::Join":[
                        "",
                        [
                            "HTTP:",
                            {
                                "Ref":"AppPort"
                            },
                            "/healthCheck"
                        ]
                    ]
                },
```

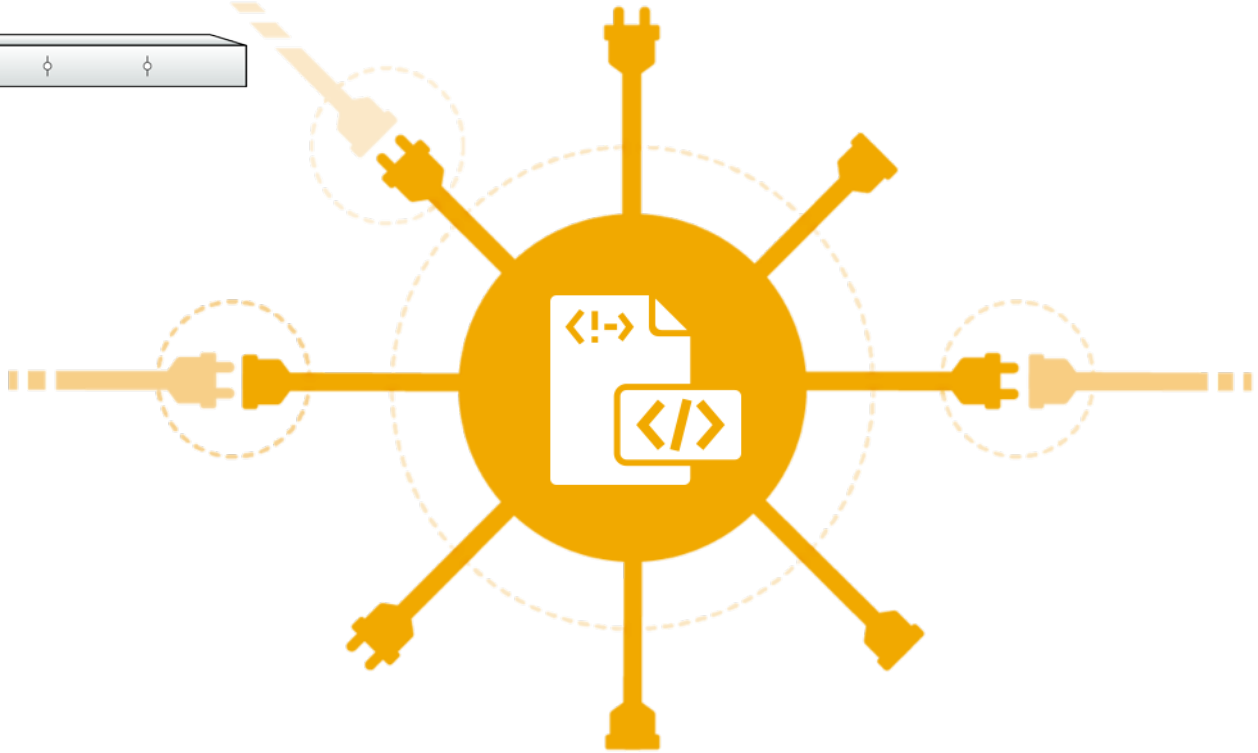
Protecting Source Code

User	Pulled	Committed	% Inactive	Avg Dist	History	TT	[redacted]
[redacted]	1397	9	58.1	11.75	view	TT:E027599[redacted]	1-Click
[redacted]	1361	4	79.2	9.08	view	TT:E027569[redacted]	1-Click
[redacted]	535		2.8	9.33	view	TT:E027655[redacted]	1-Click
[redacted]	457	11	33.0	11.34	view	TT:E027656[redacted]	1-Click

Protecting Source Code

User	Pulled	Committed	% Inactive	Avg Dist	History	TT	[redacted]
[redacted]	1397	9	58.1	11.75	view	TT:E027599[redacted]	1-Click
[redacted]	1361	4	79.2	9.08	view	TT:E027569[redacted]	1-Click
[redacted]	535		2.8	9.33	view	TT:E027655[redacted]	1-Click
[redacted]	457	11	33.0	11.34	view	TT:E027656[redacted]	1-Click

Seek vendors which embrace APIs



Source Control

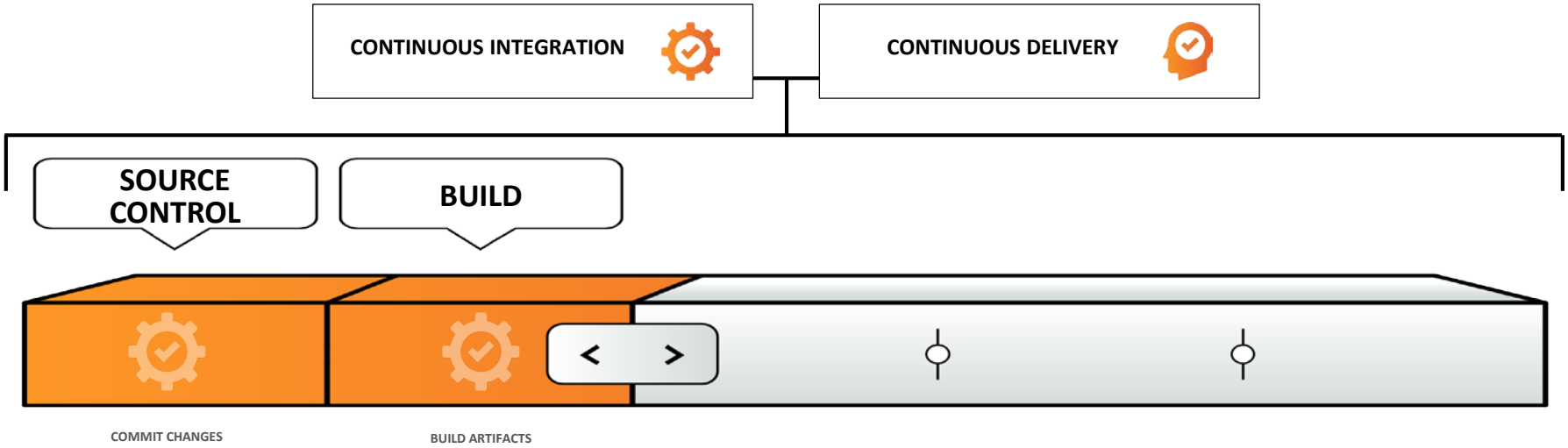
Current State

- Network and system engineers directly log into systems to make changes
- Version control for infrastructure configuration is a decoupled process
- Limited APIs awareness

Future State

- Changes are committed to source control for infrastructure and the pipeline executes the change
- Changes cannot be made without version control
- Embraces APIs

Build


CONTINUOUS INTEGRATION
CONTINUOUS DELIVERY
SOURCE CONTROL
BUILD
COMMIT CHANGES
BUILD ARTIFACTS

Is Policy A more permissive than Policy B?


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

Policy A

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

Policy B

Build Control

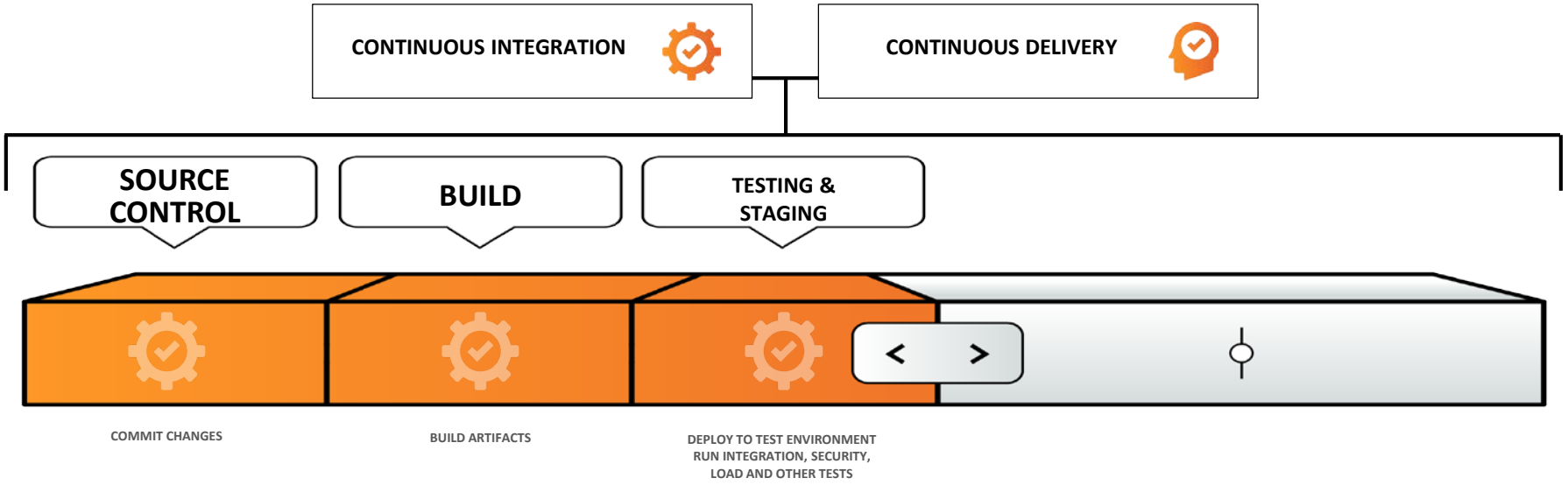
Current State

- Manual Code Review
- Manual intervention for static analysis

Future State

- Automated reasoning for formally proving security
- Automation wrapped around static analysis

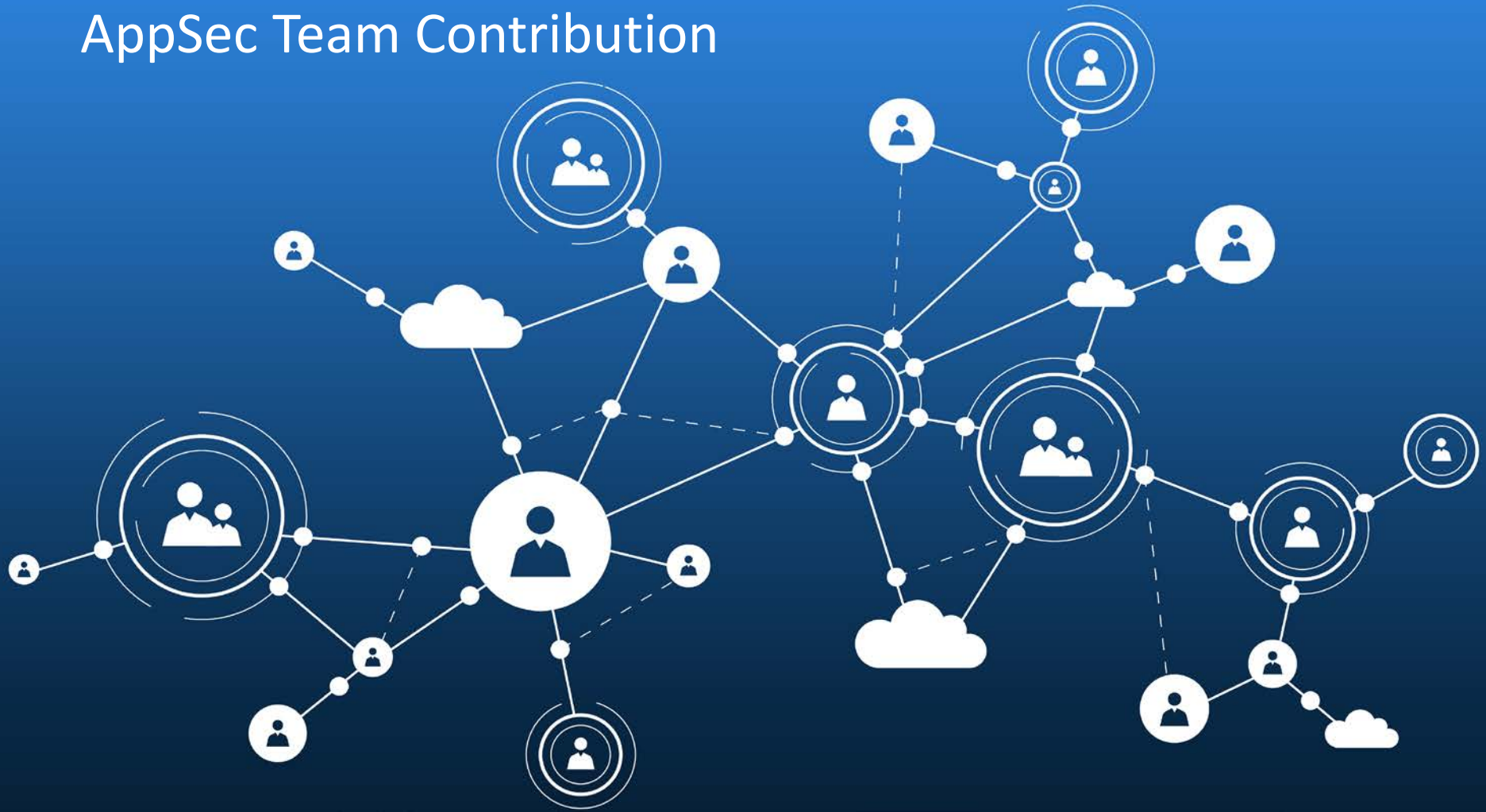
Testing & Staging



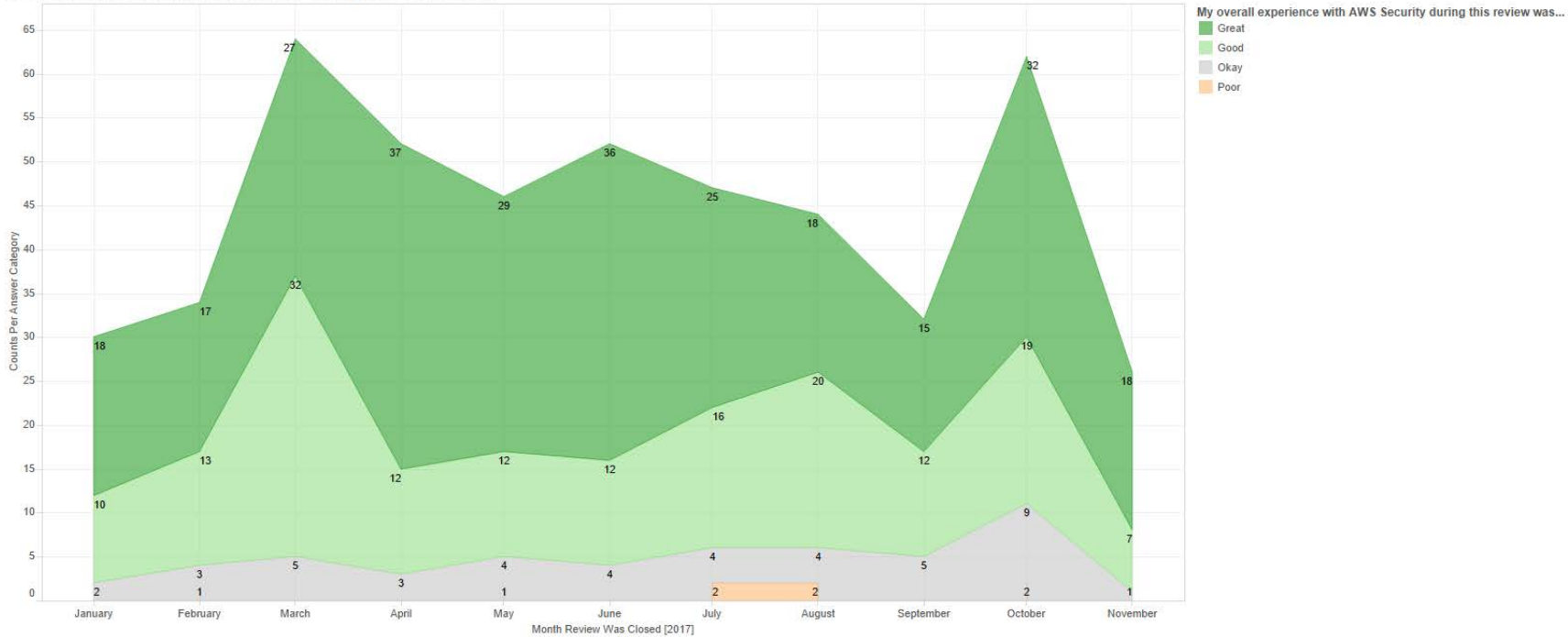
Finding Weaknesses & Defects


```
{
    "findingArns": [
        "arn:aws:inspector:us-east-1:782304905741:target/0-Um5yAZAy/template/0-h
sq25phW/run/0-szQ2RMcX/finding/0-Wq3xXMBv",
        "arn:aws:inspector:us-east-1:782304905741:target/0-Um5yAZAy/template/0-h
sq25phW/run/0-szQ2RMcX/finding/0-ZnK3JMwz",
        "arn:aws:inspector:us-east-1:782304905741:target/0-Um5yAZAy/template/0-h
sq25phW/run/0-szQ2RMcX/finding/0-xndJFsFE",
        "arn:aws:inspector:us-east-1:782304905741:target/0-Um5yAZAy/template/0-h
sq25phW/run/0-szQ2RMcX/finding/0-xWvxtocK"
    ]
}
```

AppSec Team Contribution



My overall experience with AWS Security during this review was...



Testing & Staging

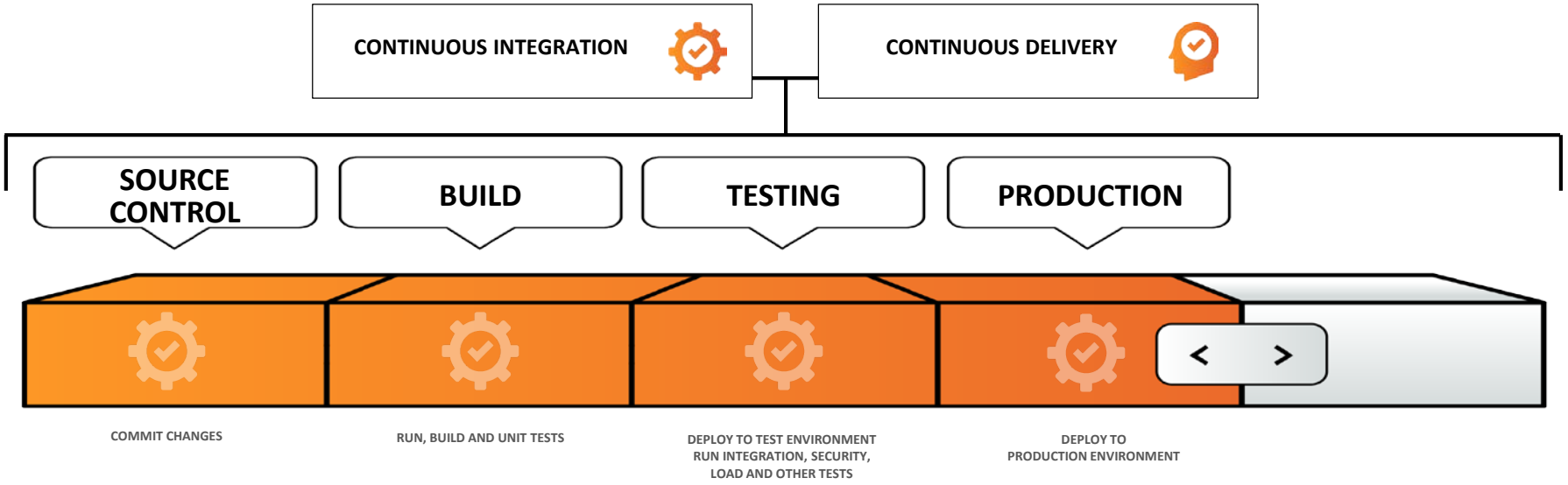
Current State

- Security assessments are manual
- Security testing is decoupled from pipelines
- Measures of AppSec team involvement are based only on risk reduction, not mutual success

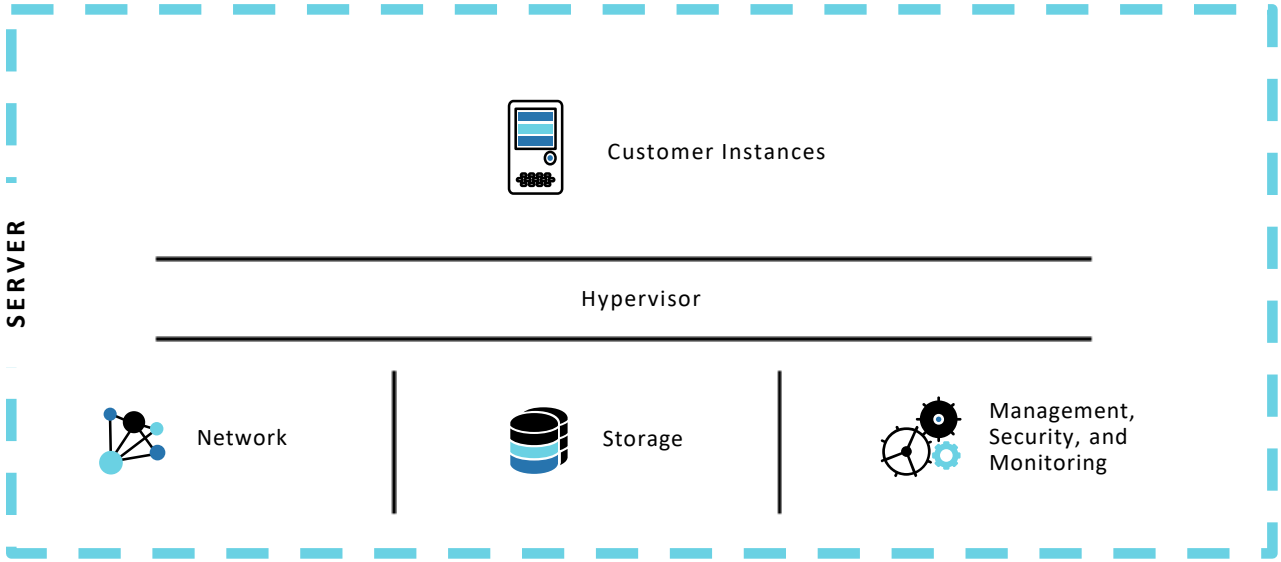
Future State

- Security assessments are coded and automated too.
- Security testing happens much closer to the time defects are created
- Feedback loops are used to ship secure code, quickly

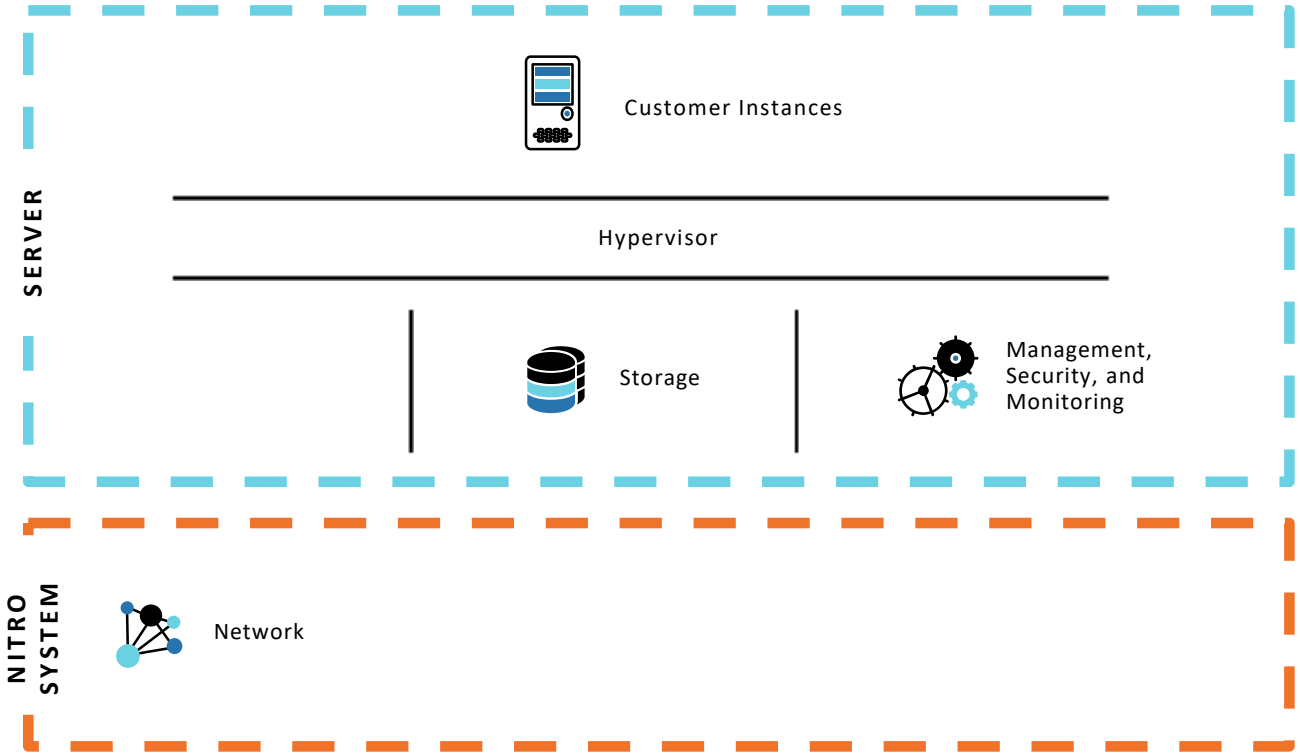
Deployment & Production



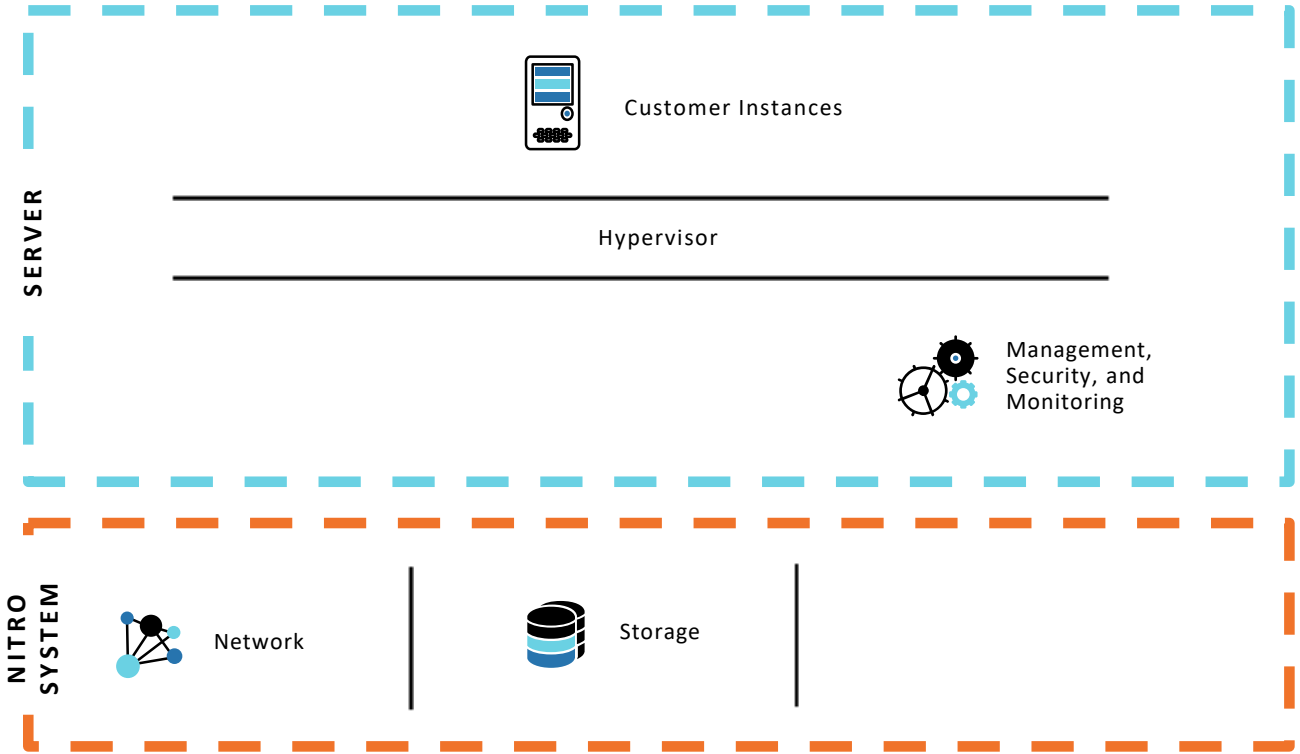
Original Amazon EC2 Host Architecture



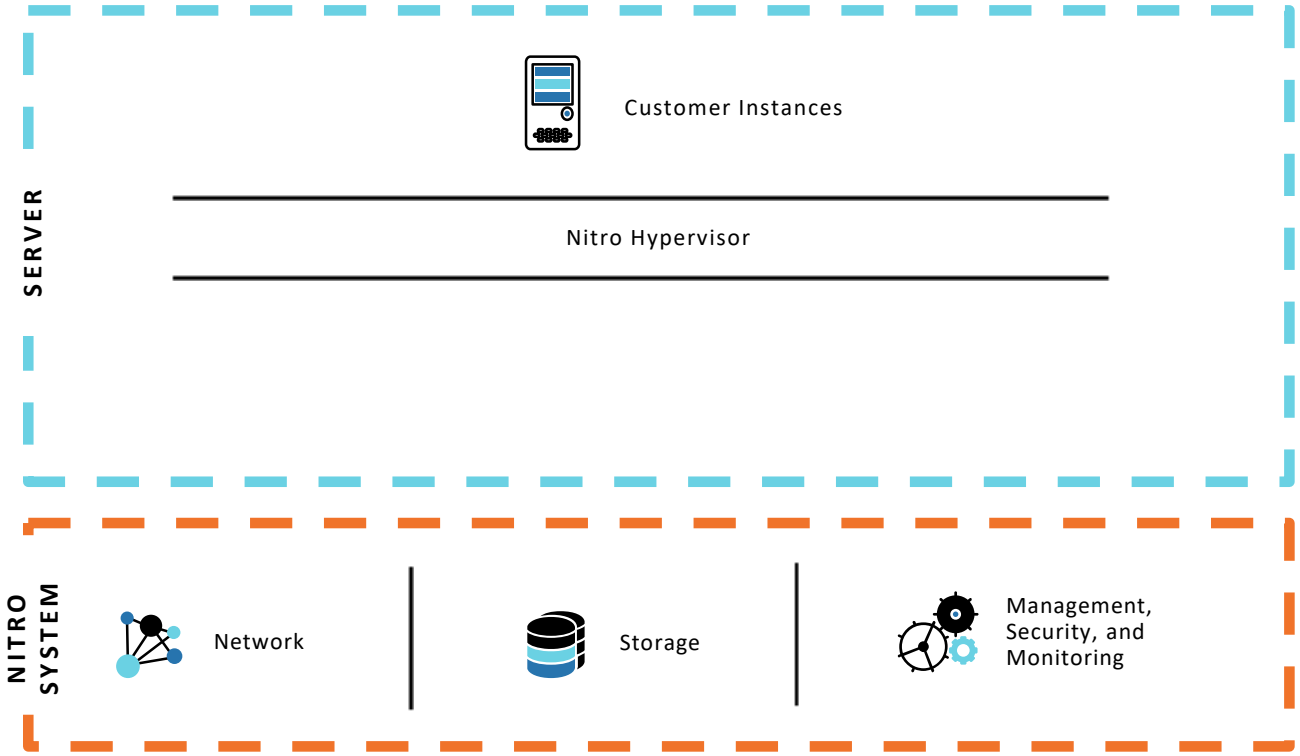
Amazon EC2 C3 Instances



Amazon EC2 C4 Instances

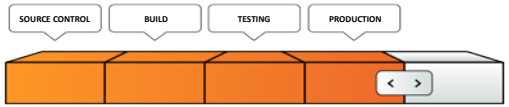


Amazon EC2 C5 Instances




SECU E SHELL
(SSH) CESS

Deployment & Production Summary



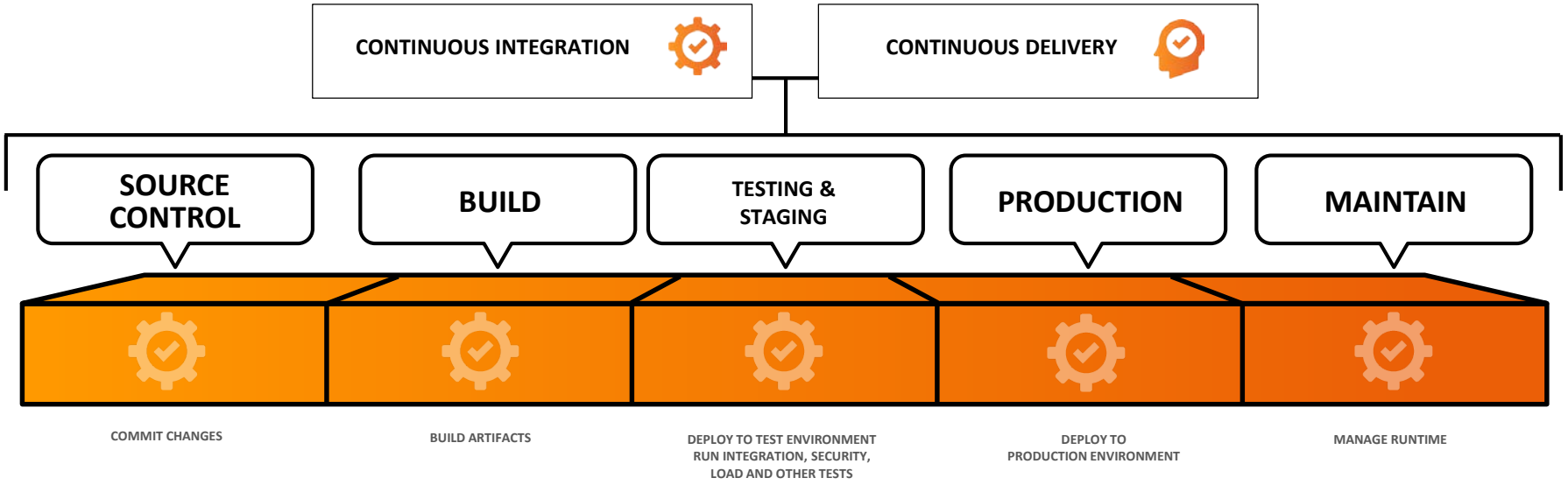
Current State

- Persistent shell access to production

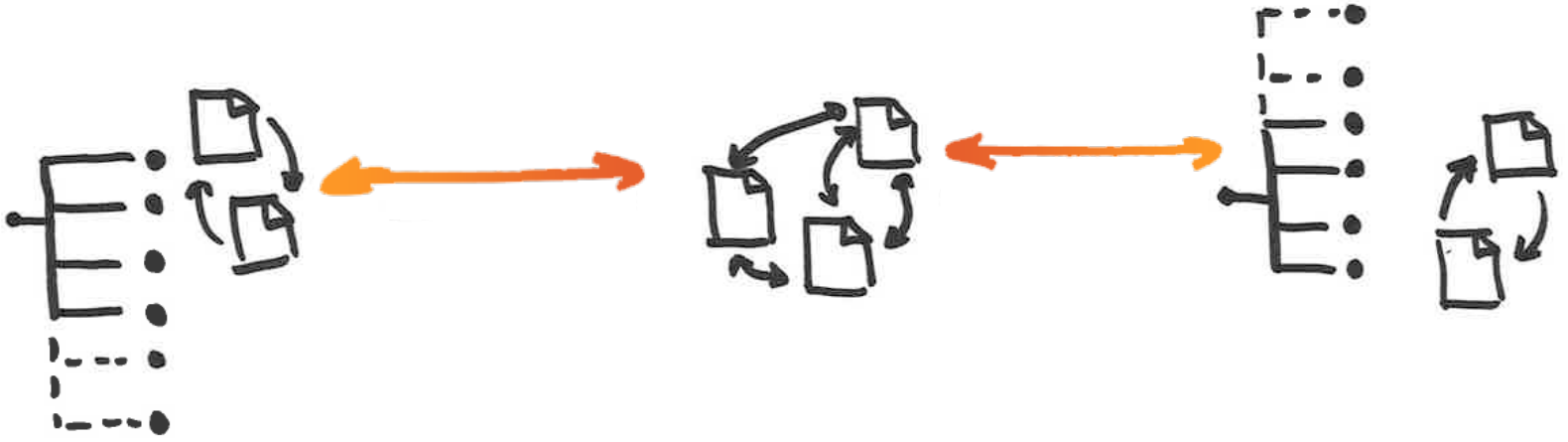
Future State

- Runtime automation, runbooks that constrain and reduce shell access
- Rotational access where required
- Code is deployed to production via pipelines, not over walls.

Maintaining Runtime Environment



Use ML and Scaled Services

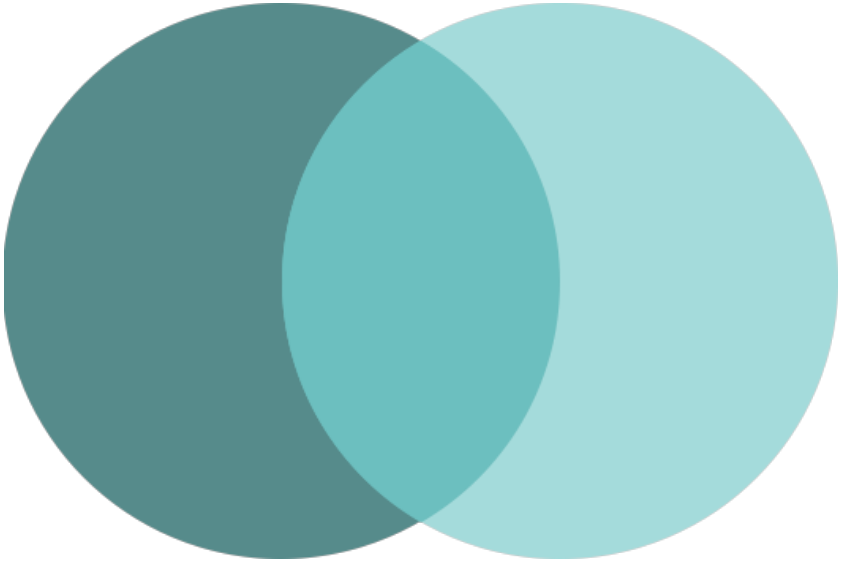


IP Reputation Service

Log Processing Fleets

DNS Reputation Service

Using NLP and ML together



Understand your data
Natural Language Processing (NLP)

Understand data access
Predictive User Behavior Analytics (UBA)

Content Classification with NLP

PII and personal data

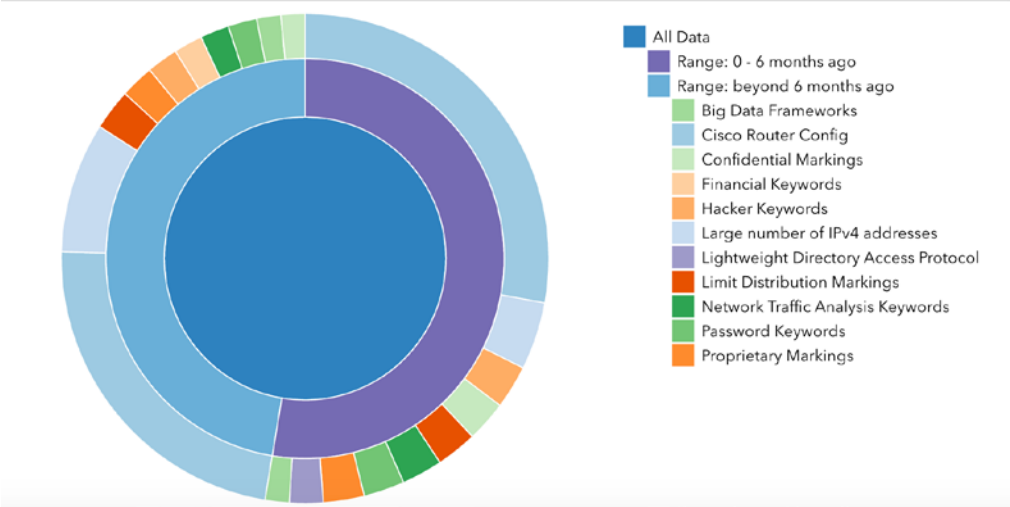
Source code

SSL certificates, private keys

iOS and Android app signing keys

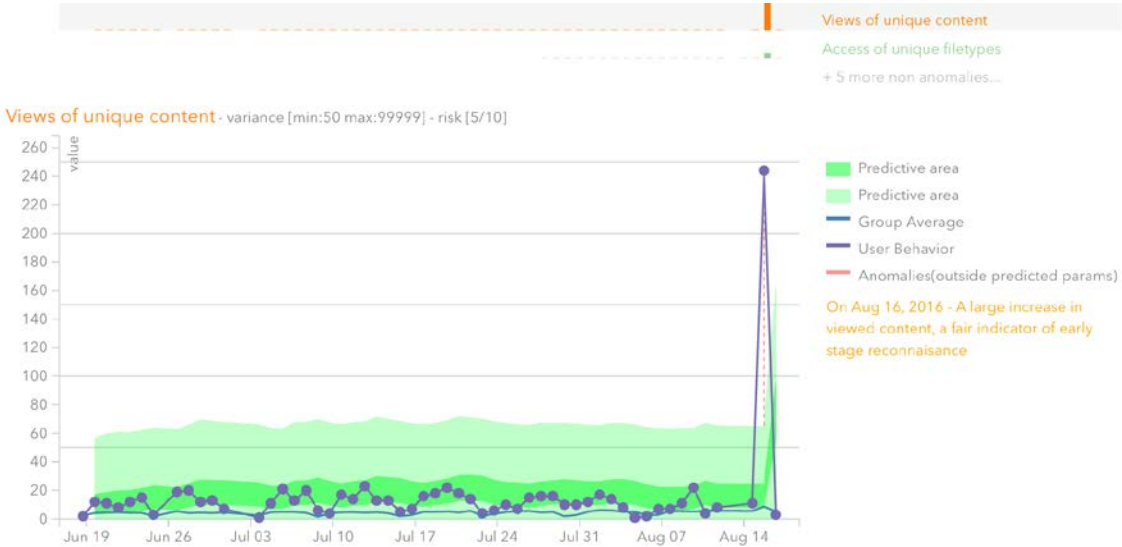
Database backups

OAuth and Cloud SaaS API Keys



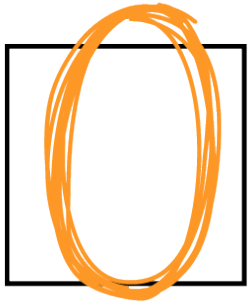
Use ML and Scaled Services

- Use behavioral analytics to baseline normal behavior patterns
- Contextualize by value of data being accessed



Tools we use: COEs

THIS TEAM HAS WORKED



DAYS

WITHOUT AN INCIDENT

Maintain Runtime Environment Summary

Current State

- Inability to scale reputation-based services.
- Difficult to classify data and detect anomalies in access.

Future State

- Leverage cloud services for computationally expensive capabilities.
- Apply NLP and machine-learning together to classify sensitive data and detect anomalies.
- Focus on COEs

Call to Action – Do Try This at Home

- In your company, deeply understand how software is created and shipped. Sit security team members with a development team for as many days as you can (and not just the appsec team) (1-2 months).

Call to Action – Do Try This at Home

- In your company, deeply understand how software is created and shipped. Sit security team members with a development team for as many days as you can (and not just the appsec team) (1-2 months).
- Catalog the controls and visibility into CI/CD pipelines. That's where change management and control happens now (1-3 months).

Call to Action – Do Try This at Home

- In your company, deeply understand how software is created and shipped. Sit security team members with a development team for as many days as you can (and not just the appsec team) (1-2 months).
- Catalog the controls and visibility into CI/CD pipelines. That's where change management and control happens now (1-3 months).
- Begin to document every instance of human interaction with systems that process data. Let engineering & operations teams drive this goal. (1-6 months).

Call to Action – Do Try This at Home

- In your company, deeply understand how software is created and shipped. Sit security team members with a development team for as many days as you can (and not just the appsec team) (1-2 months).
- Catalog the controls and visibility into CI/CD pipelines. That's where change management and control happens now. Set clear goals with owners to harden the pipeline (1-3 months).
- Begin to document every instance of human interaction with systems that process data. Let engineering & operations teams drive this goal. (1-6 months).
- Set and achieve a goal to reduce human access to systems that process sensitive data by 80% (1-2 years).

Call to Action – Do Try This at Home

- In your company, deeply understand how software is created and shipped. Sit security team members with a development team for as many days as you can (and not just the appsec team) (1-2 months).
- Catalog the controls and visibility into CI/CD pipelines. That's where change management and control happens now. Set clear goals with owners to harden the pipeline (1-3 months).
- Begin to document every instance of human interaction with systems that process data. Let engineering & operations teams drive this goal. (1-6 months).
- Set and achieve a goal to reduce human access to systems that process sensitive data by 80% (1-2 years).
- Set and achieve a goal to drive workload deployment from source code. Catalog the % of workloads that are built on automation vs. those built with manual steps (1 year).

More Info

AWS Security Twitter: @AWSSecurityInfo

AWS Security Blog: aws.amazon.com/blogs/security/