

RSA® Conference 2018

San Francisco | April 16 – 20 | Moscone Center



#RSAC

SESSION ID: CSV-R04

PRAGMATIC SECURITY AUTOMATION FOR CLOUD

Rich Mogull

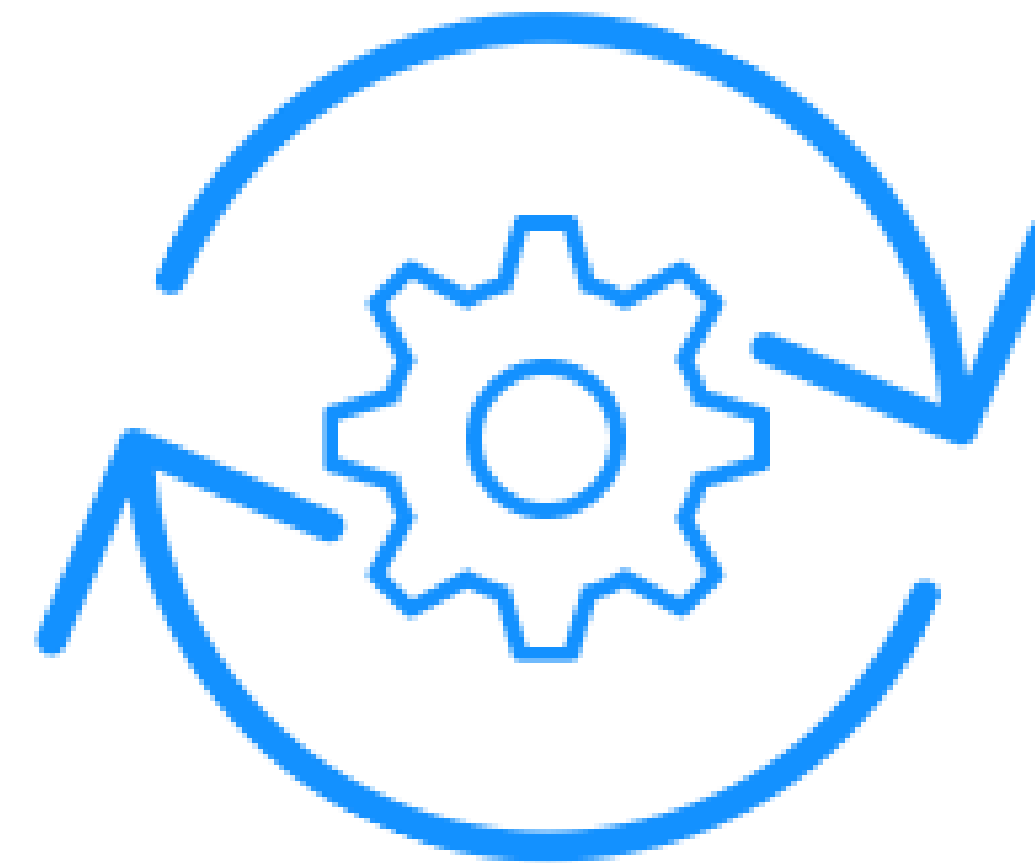
Analyst/VP of Product
Securosis/DisruptOPS
rmogull@disruptops.com
@rmogull



Cloud is Fundamentally Different

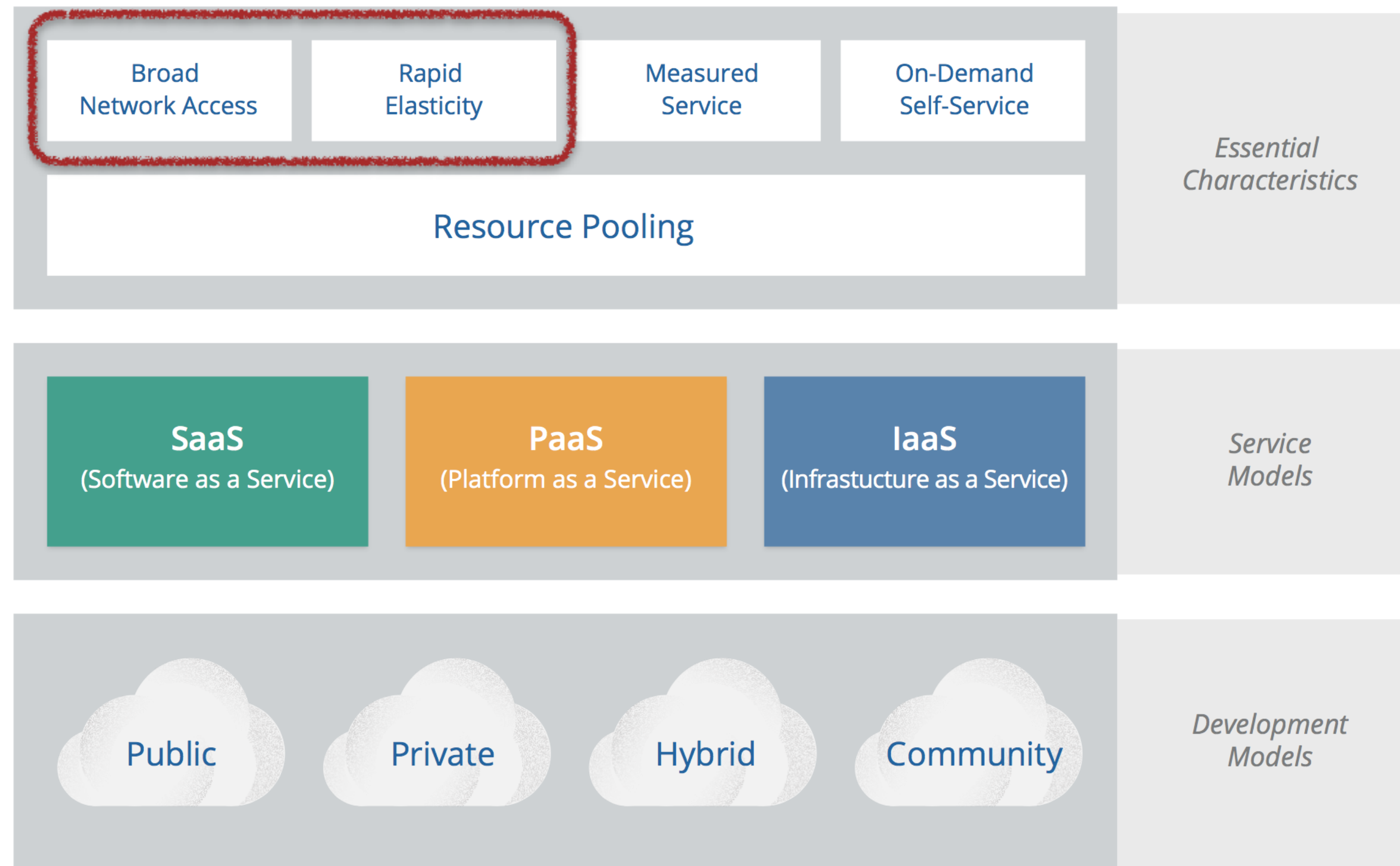


Abstraction



Automation

Automation is Inherent

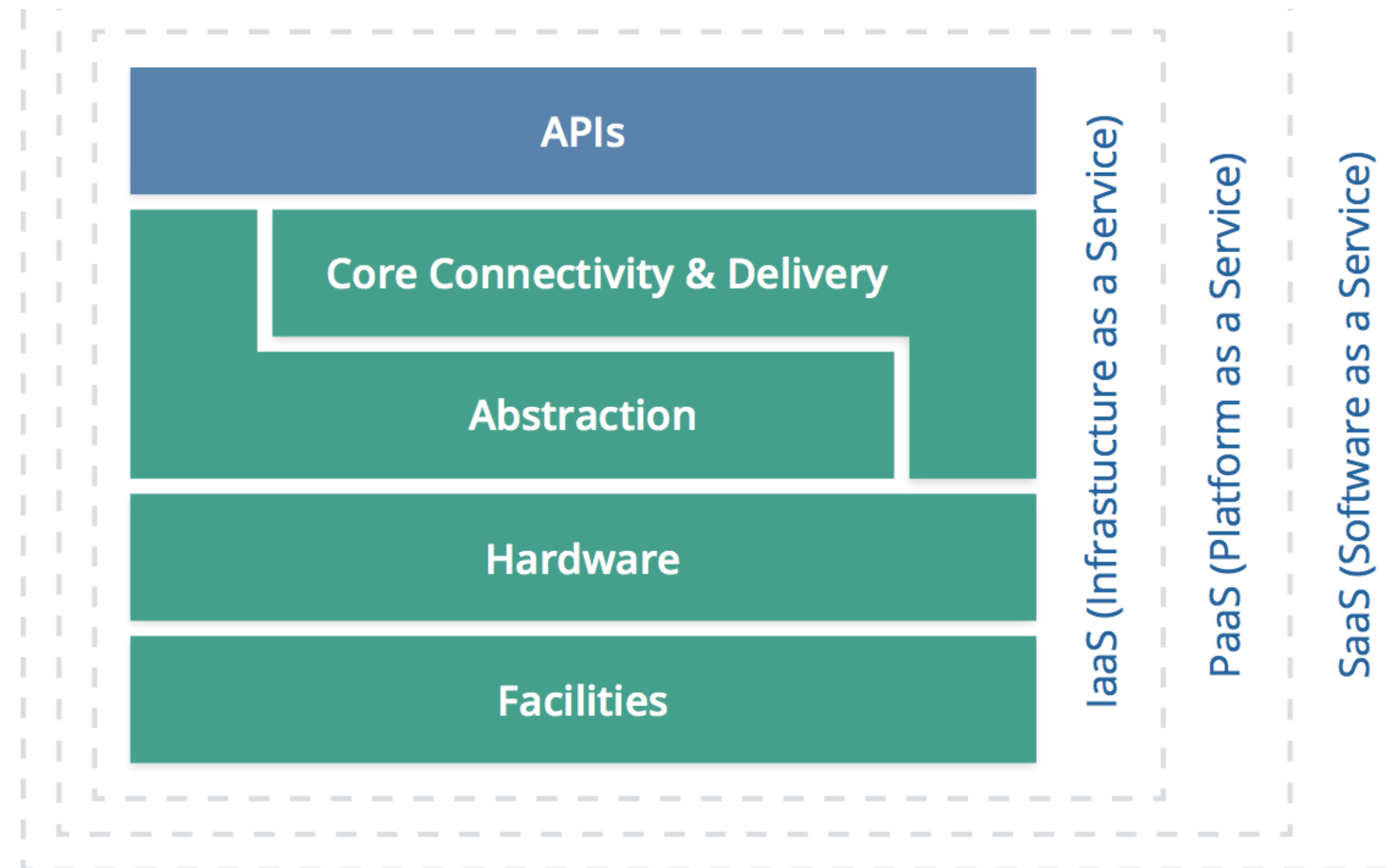


The NIST Model (courtesy the CSA)

APIs are Ubiquitous



Cloud Security Alliance IaaS
Reference Model



Cloud Security Must Be Cloud Native



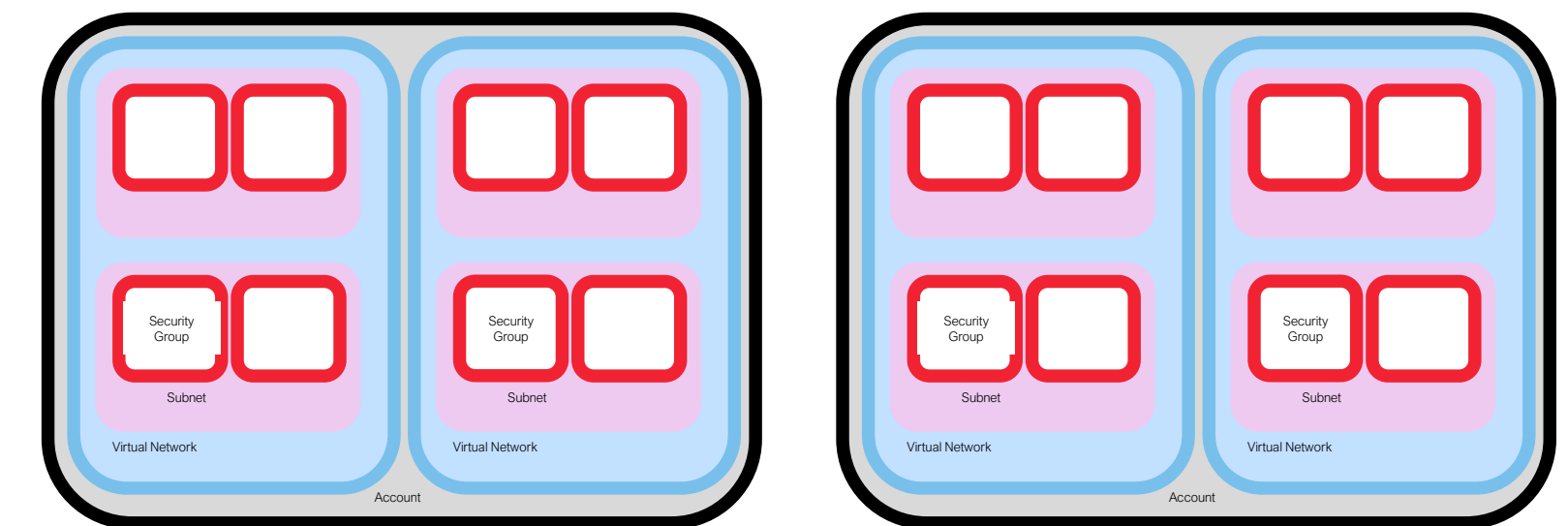
Management Plane



Volatility/Velocity



Distribution/Segregation



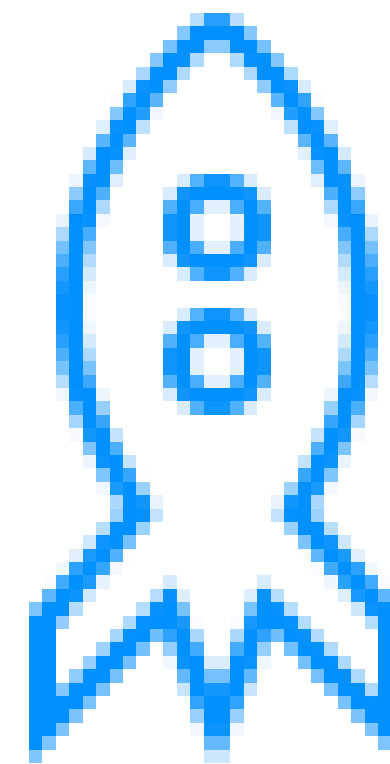
The Categories



Guardrails

Continuously assess and enforce operational and security policies

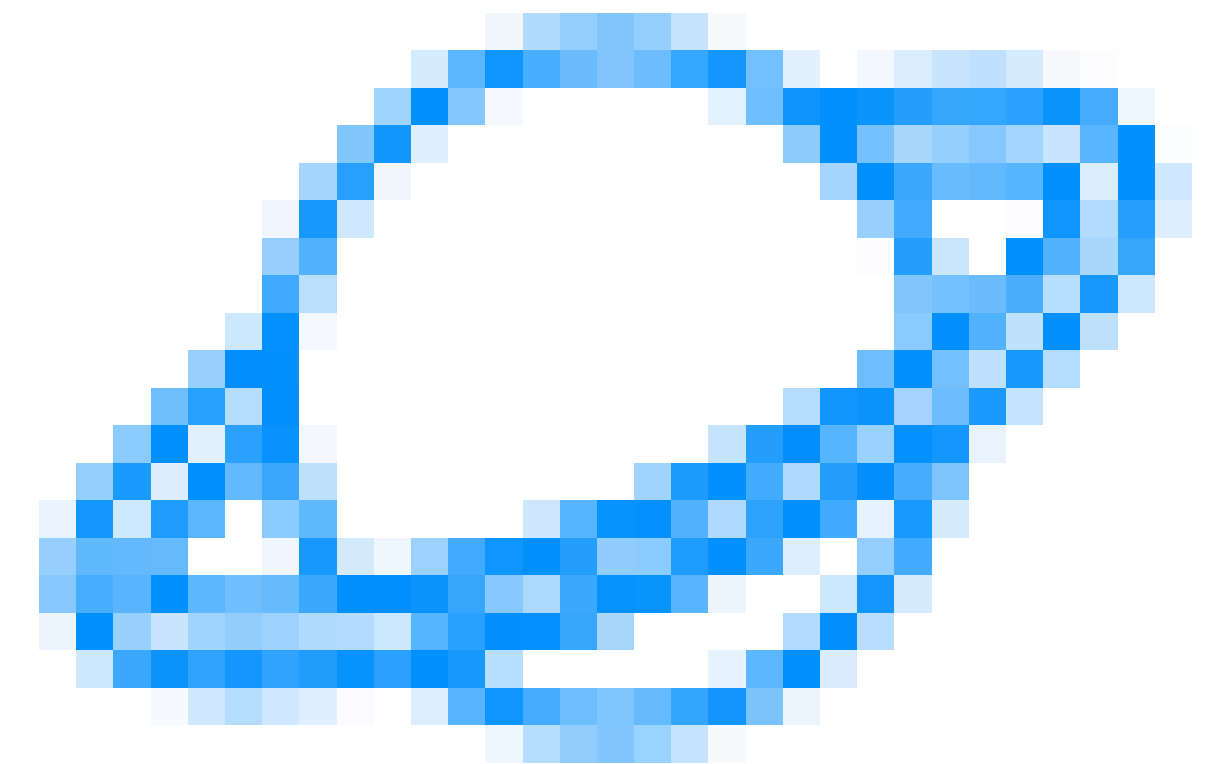
Fix security group or S3 misconfigurations



Workflows

Streamline and accelerate IT operations and security through automated workflows

Incident response



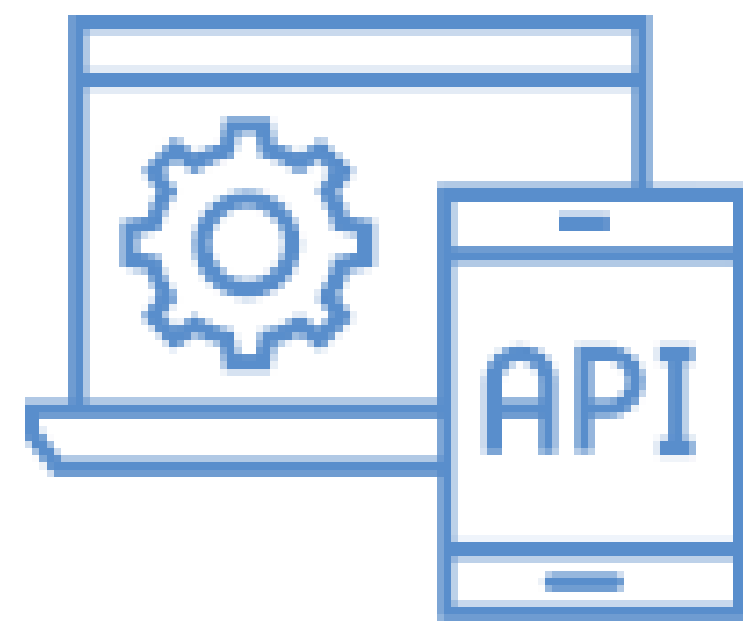
Orchestrations

Empower new capabilities through advanced orchestration of infrastructure, operations, and security

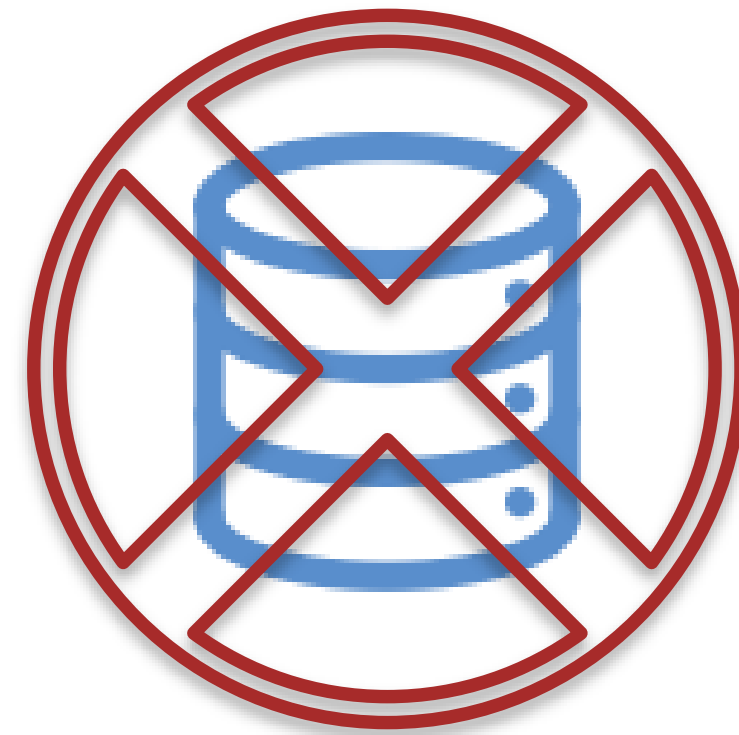
Automatic WAF insertion and configuration



The Principles



Software
Defined
Security



Stateless
Security



Event Driven
Security



Continuous
Feedback
Loops

The Foundation



Cloud Service Provider

- ▣ API and full administrative activity logging
- ▣ Events/triggers/rules
- ▣ Function as a Service (Serverless)
- ▣ Notification service

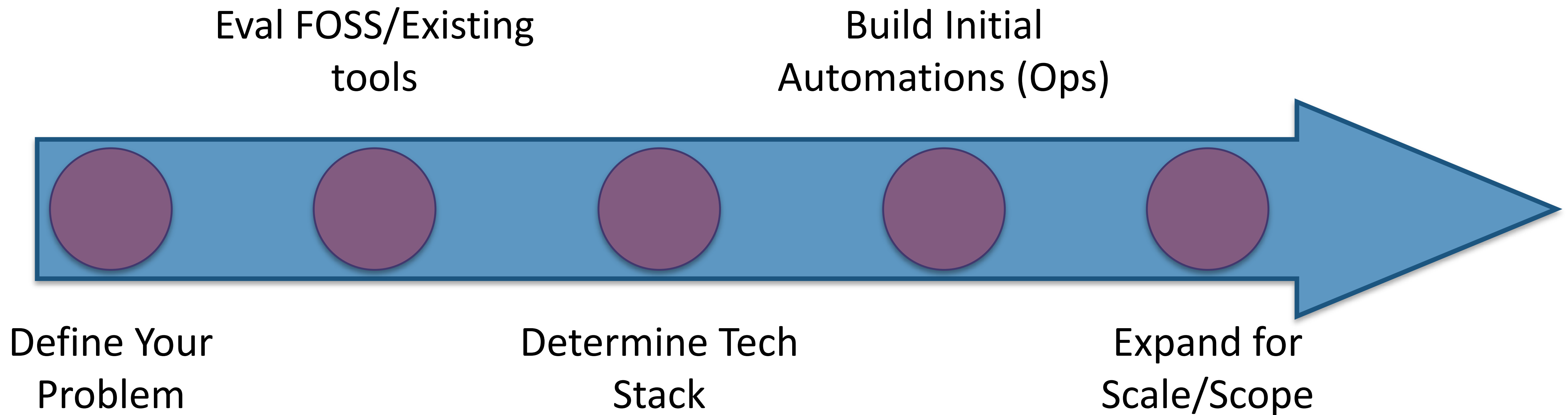
Critical Capabilities

Cloud Consumer (you)

- ▣ Continuous Integration Pipeline
- ▣ Version control repository
- ▣ Full IAM access to accounts/subscriptions/projects
- ▣ Security development team (person)



The Process



Things We Are Skipping (for time)

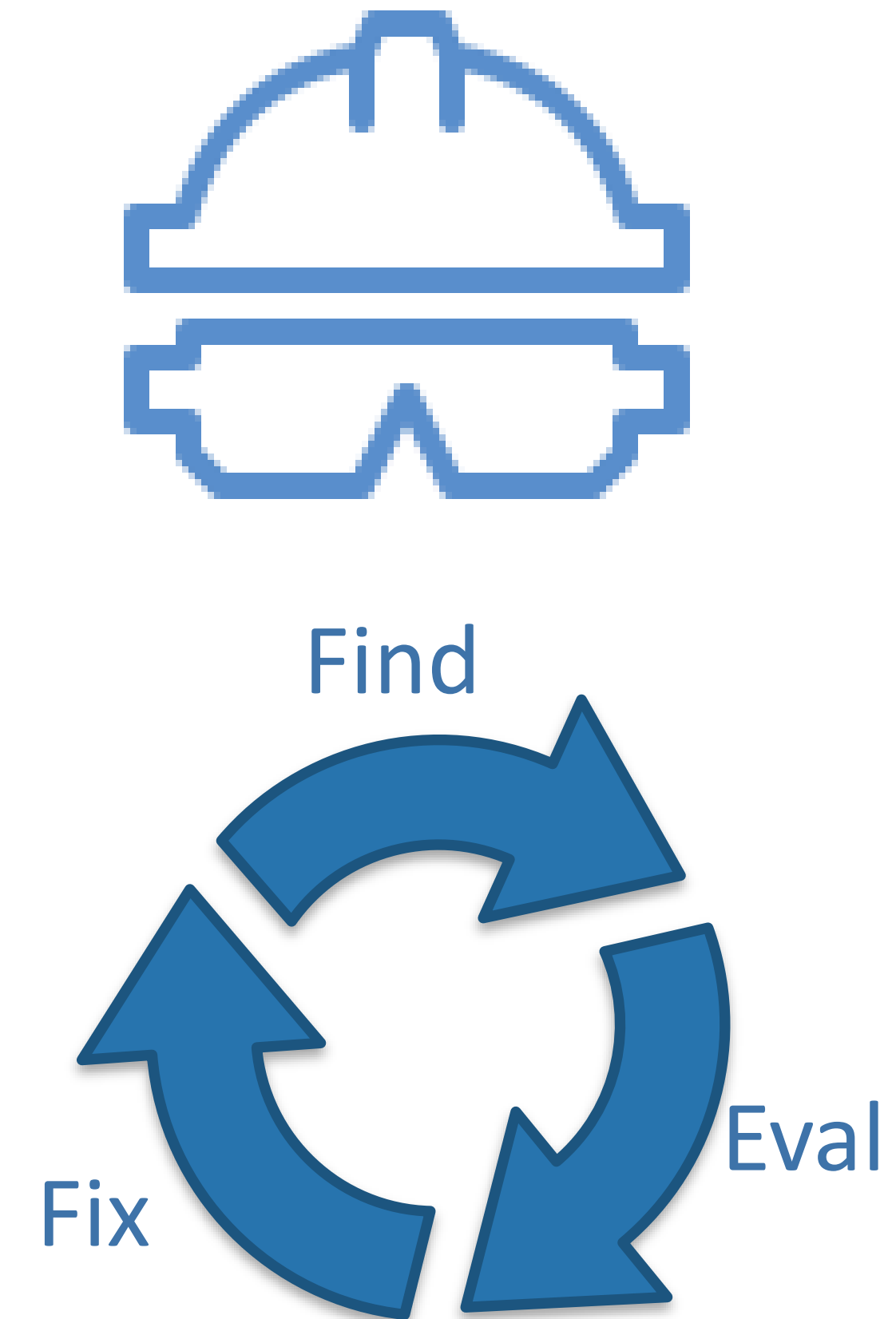


- How to configure all the core monitoring/logging
- Setting up IAM and permissions
- The details of implementation on Azure and GCP
 - We will list the core capabilities, but can't cover all 3 with real examples in 45 minutes

What's a Guardrail?



- Define and set limits
 - Can be “allow” or “deny”
- Find deviations
 - Assessment or event based
- Evaluate the issue
- Fix/remediate
 - Automatically or manually depending on rules



Example Guardrails



- If you find a public S3 bucket, restrict it to our known network addresses
 - Unless it is approved or tagged
- Don't allow internal security groups with all ports and protocols open in Prod
 - But allow in Dev
- Require MFA for API access for any user that needs MFA for console access
- Create our baseline IAM policies and roles for all new accounts
 - Based on the environment
- Validate that monitoring and alerting is properly configured
 - And fix if not
- Disable access keys that haven't been used in 90 days
- Find instances with an IAM role that allows power user or greater access via API
 - Restrict the privileges
- Identify all cross-network peering from accounts we don't own
 - Then check the security group permissions

What Makes a *Good* Guardrail?



- Accounts for different environments
 - At least Dev vs. Prod
- Handles exceptions
 - And is capable of remembering them
- Understands state and context
- Doesn't bog down the alert queue
- Can remediate automatically
 - Either completely, or after manual approval
- Ops communications/notifications
- Education, not Blamification

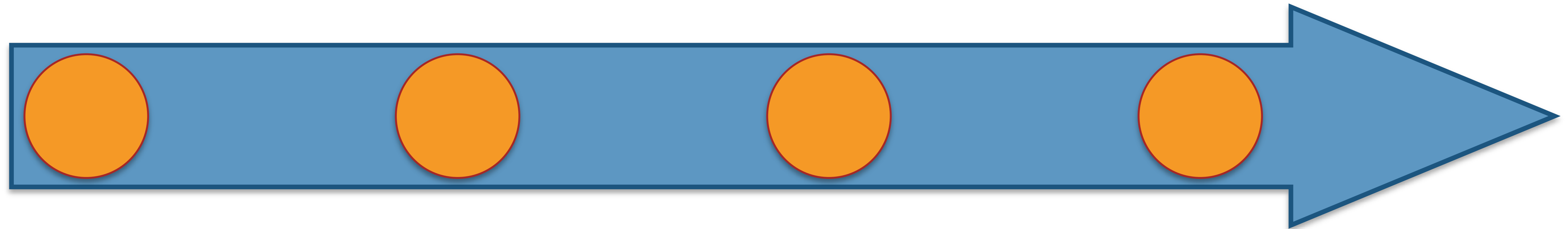


Building a Guardrail



Add Filters

Add Actions
And Targets



Define
Criteria/Issues

Set Triggers

Our Guardrail



- Criteria/Issues
 - All instances with port 22 open to the 0.0.0.0/0 (the Internet)
- Filters
 - Region is us-west-2p (could be VPC/tag/etc)
- Trigger
 - Time = every 5 minutes
- Action
 - Restrict to known IP range

Step 1: Create rule

Create rules to invoke Targets based on Events happening in your AWS environment.

Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

☐ Event Pattern ⓘ ☒ Schedule ⓘ

☒ Fixed rate of

☐ Cron expression

[Learn more](#) about CloudWatch Events schedules.

► Show sample event(s)

* Required

Demo

Our Event-Driven Guardrail



- Criteria/Issues
 - New inbound security group rule added
- Filters
 - IAM user, VPC, Tag
- Trigger
 - API event (CloudTrail)
- Action
 - Reverse + Notify



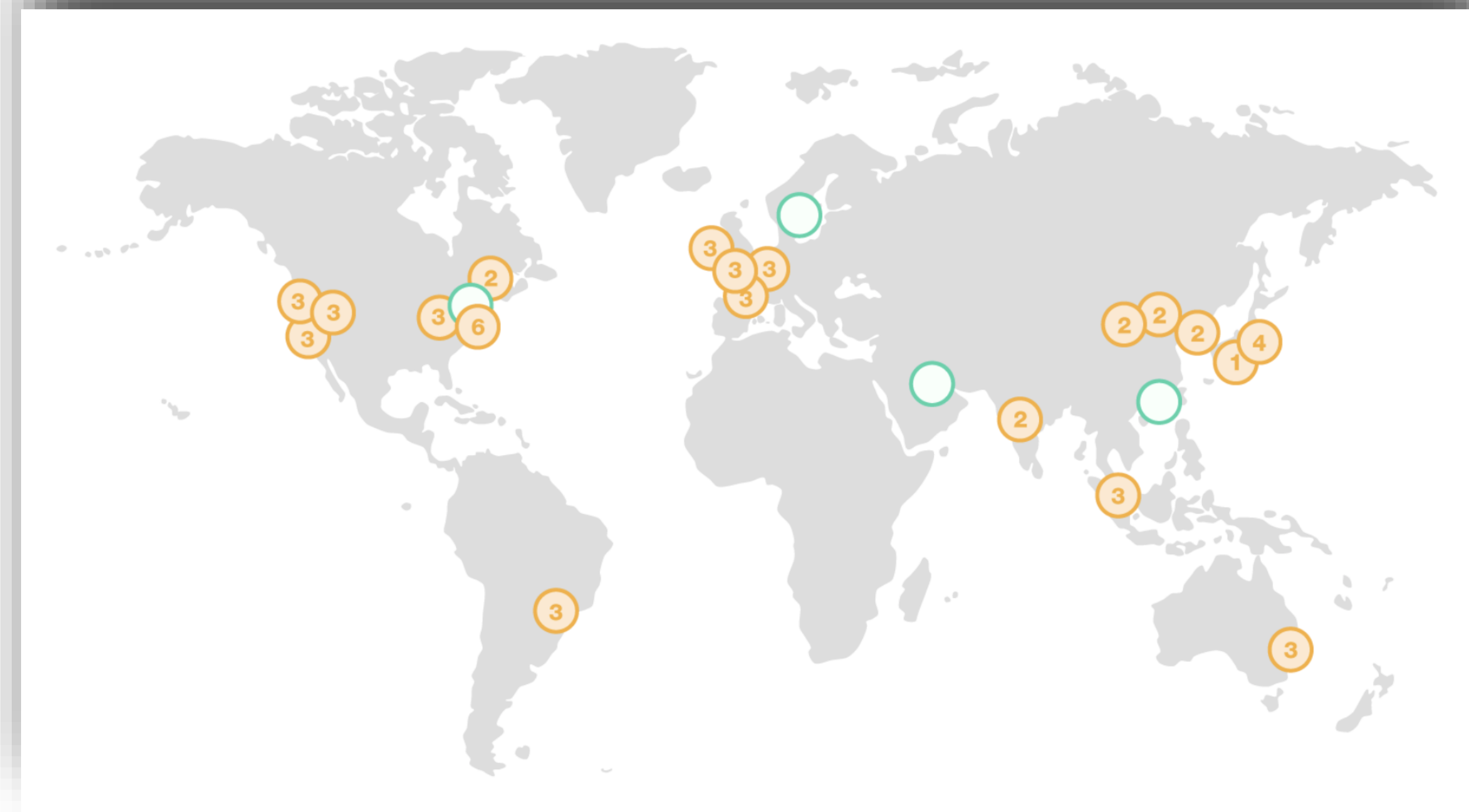
```
File Edit Find View Goto Tools Window
Environment
  fixSecurityGroup
    lambda_function.py
lambda_function.py
1 from __future__ import print_function
2
3 # A demonstration AWS Lambda function to revert a security group change based on a CloudWatch event.
4 # By Rich Mogull and Securosis, released under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International license.
5 # http://creativecommons.org/licenses/by-nc-sa/4.0/ ***** https://securosis.com
6
7 # This lambda function will reverse any security group changes when triggered by a CloudWatch event
8 # that shows an ingress change. It *does not* work for outbound changes, but as you can see
9 # if you review the code it could easily be modified for that.
10 # The demonstration code also includes various conditional options. If you don't use one of these
11 # this will apply to every change in your account. To use them, modify the parameters and then
12 # move the function call into the conditional block.
13
14 import json
15 import urllib
16 import boto3
17
18 print('Loading function')
19
20 ec2 = boto3.client('ec2')
21
22
23 def lambda_handler(event, context):
24     # dump the raw event for log purposes
25     print("An unauthorized security group change was detected and will be remediated. The event details are:")
26     print("-----")
27     print("Received event: " + json.dumps(event, indent=5))
28     print("-----")
29
30     # Only execute if the change was in a certain region
31     if event["detail"]["awsRegion"] == "us-west-2":
32         print("Region is us-west-2")
33
34     # Only execute if the change *was not* from a designated admin account
```

Demo

Expanding to Enterprise Scale



- Hitting all 14 regions simultaneously
- Multiplex
- Central event stream
- Queues/SNS
- AuthN/AuthZ

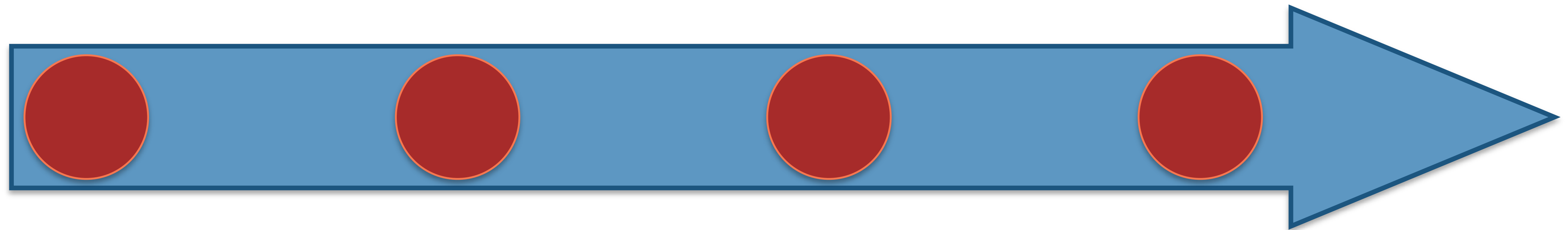


Building a Workflow



Determine Inputs

Modularize Code



Define Steps

Choose Execution Model

Can be built on Guardrails and support Orchestrations

Our Workflow



- Steps (Incident Response)
 - Collect metadata (before we change it)
 - Quarantine on the network and in AWS
 - Snapshot all storage and attach for forensics
 - Analyze
- Inputs
 - Instance ID
- Execution Model
 - Command line (container or remote)
- Modularize Code
 - Classes for analyze vs. respond
 - All methods reusable

```
SecuritySquirrel — ruby — 83x26

Enter Instance ID:i-3dbd9f09
Metadata for i-3dbd9f09 appended to ForensicMetadataLog.txt

Quarantining i-3dbd9f09...
i-3dbd9f09 moved to the Quarantine security group from your configuration settings.

Tagging instance with 'IR'...
Instance tagged and IAM restrictions applied.

Identifying attached volumes...
Volume vol-2d3edb21 identified; creating snapshot
Snapshots complete with description: IR volume vol-2d3edb21 of instance i-3dbd9f09
at 2014-02-20 11:47:32 -0700
Volume vol-6212f26e identified; creating snapshot
Snapshots complete with description: IR volume vol-6212f26e of instance i-3dbd9f09
at 2014-02-20 11:47:32 -0700

A forensics analysis server is being launched in the background in with the name
'Forensics' and the snapshots attached as volumes starting at /dev/sdf
(which may show as /dev/xvdf). Use host key rmogull-oregon for user ec2-user

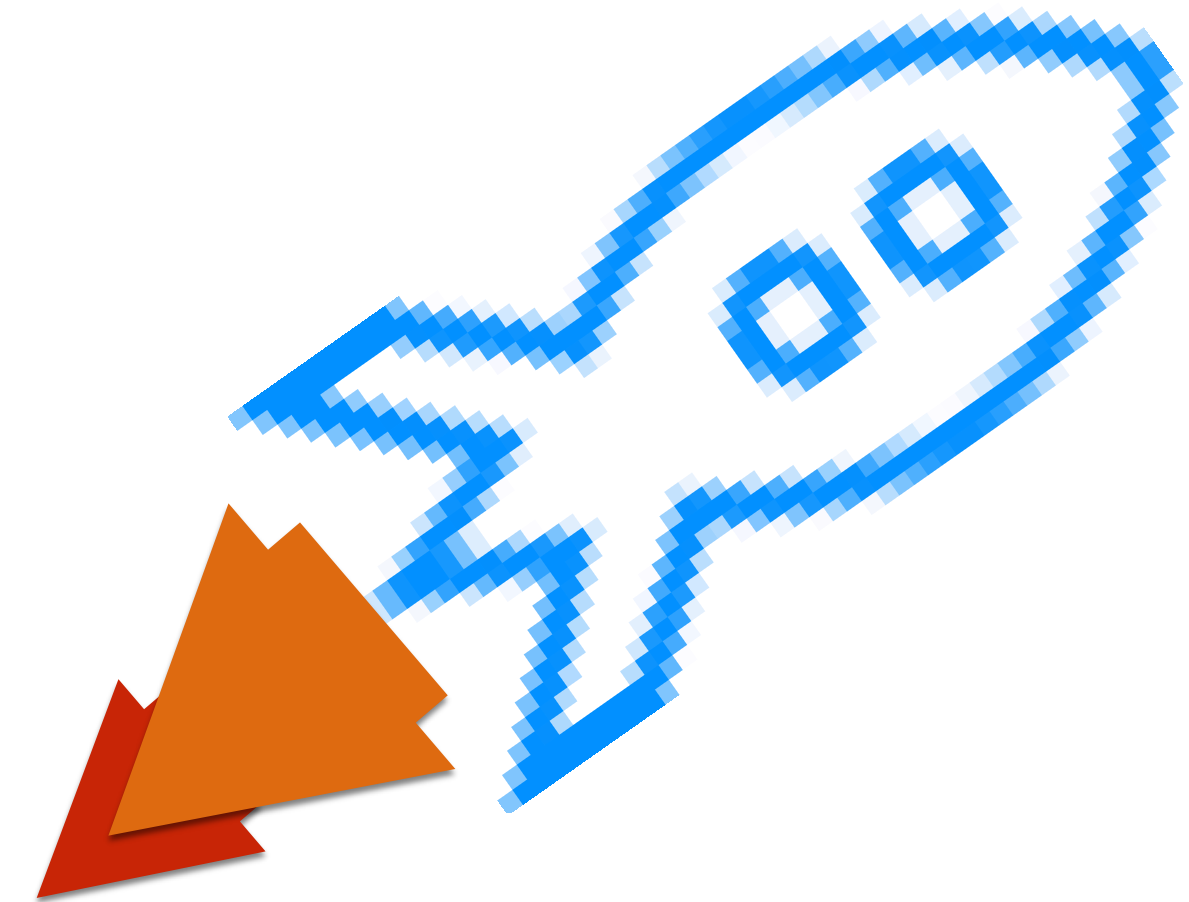
Press Return to return to the main menu
█
```

Demo

Workflows Advice



- Workflows are to speed up common, manual tasks
 - Guardrails are for automated enforcement
 - The line between a guardrail action and an Workflows is often thin
- Execution environment matters
 - Lambda vs. containers vs. your laptop
- Use your pipeline
 - Continuous integration servers (Jenkins) make great platforms for repeat automation, not just security testing
- Make a static console
 - E.g. S3 + API Gateway + SQS

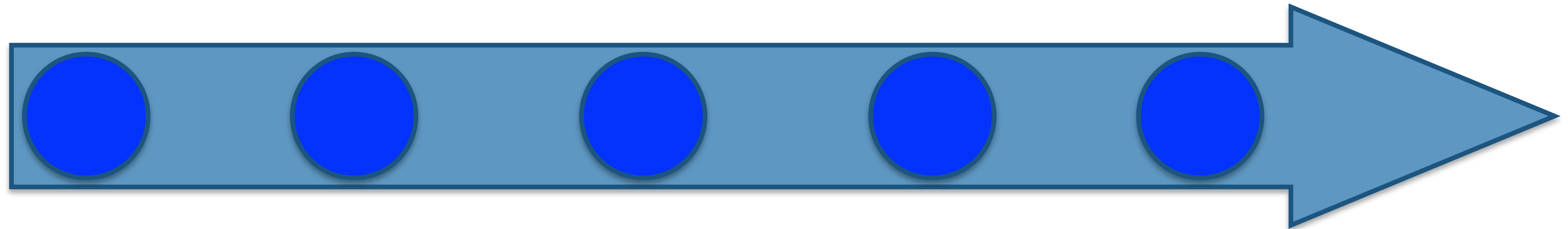


Building a Orchestration



Locate SDK if available

Modularize



ID apps and APIs

Consider
flow/value

Integrate in code

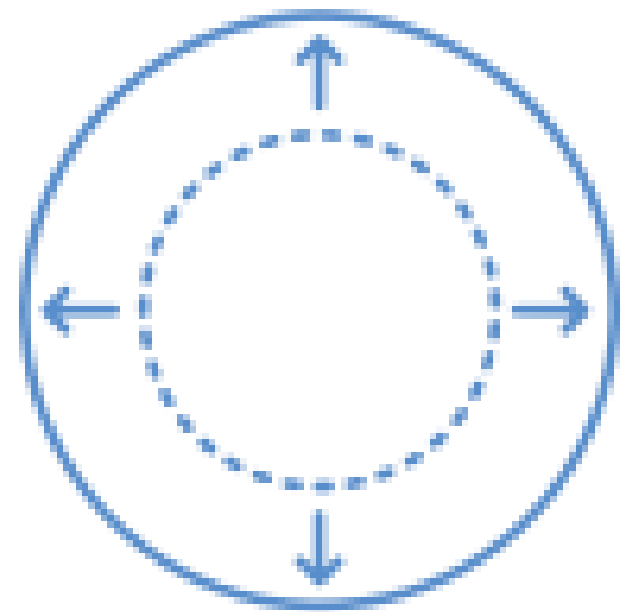
Our Orchestration



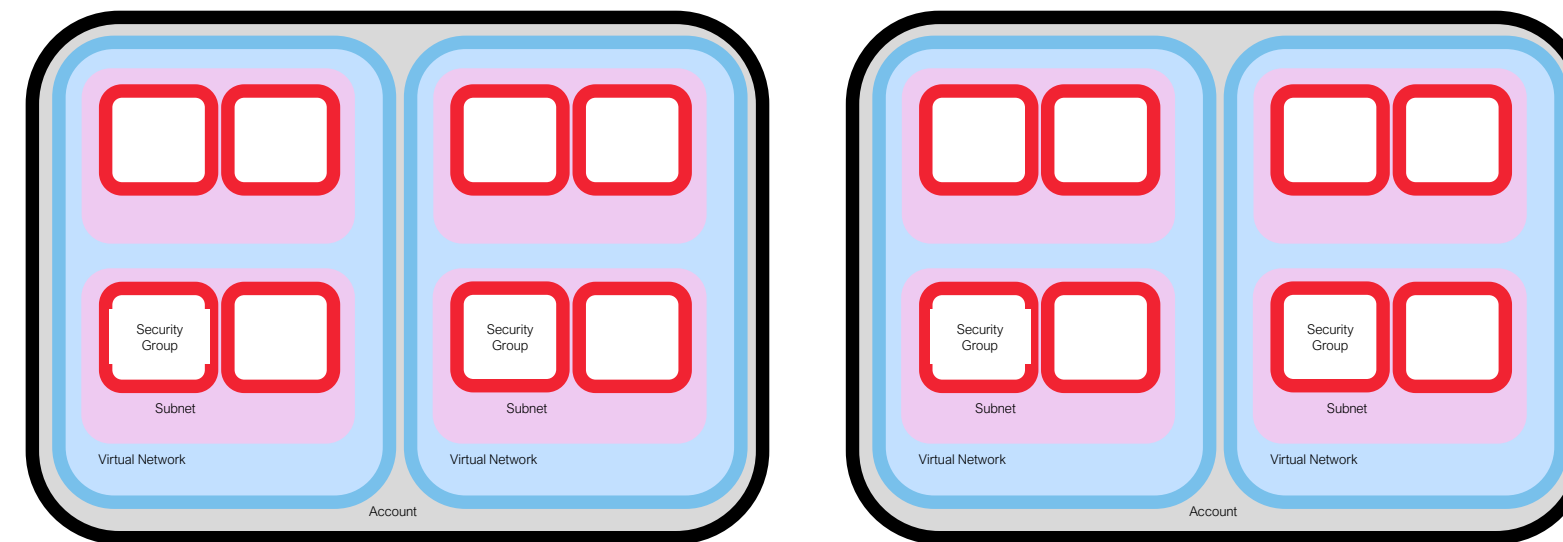
- Apps/API
 - EC2 + Route 53 + Incapsula
- SDK
 - AWS Ruby + REST client
- Flow/Value
 - ID public web servers -> determine DNS -> check WAF -> add WAF
 - Limit: default AWS domain names
- Modularize
 - Find web instances, ELBs
 - Change DNS, add Incapsula
- Integrate into code
 - See video

```
2. ruby
This workflow scans your Amazon deployment to identify web servers (instances and elastic load balancers) not protected with the incapsula Web Application Firewall.
If unprotected servers have domain names managed by Route53, you will have the option to add Incapsula protection.
Press Return to begin:
█
```


Complexities



Scaling



Multiple Accounts

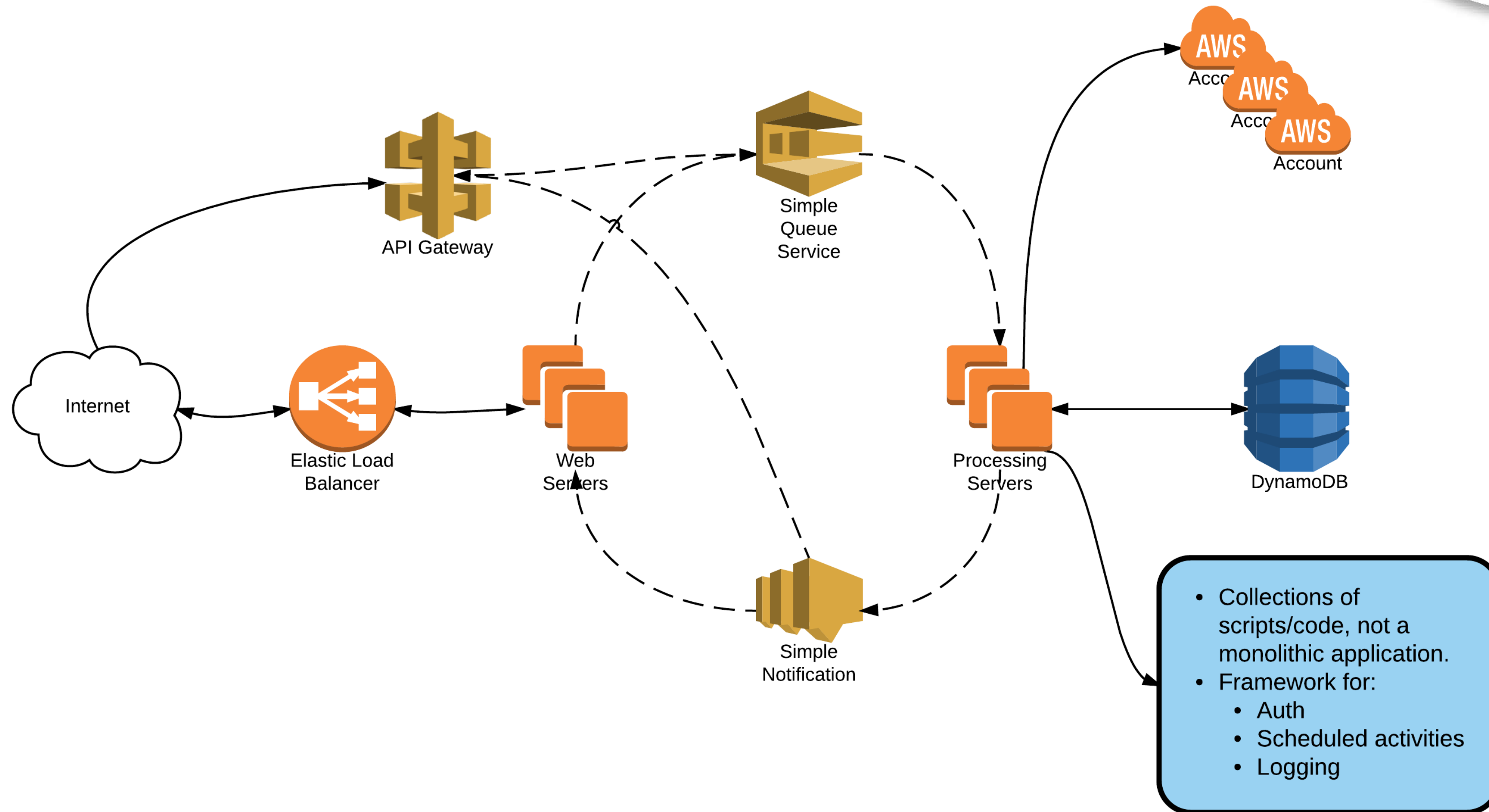


Multiple Providers



Circuit Breakers

Architecting For Enterprise Scale



Where to Start



- Start with something simple
 - Build it in one account/subscription/project
 - Event + Notification is super easy to start
 - Then go with your first FaaS
 - Desktop first, then FaaS for execution environment
- Build a library
 - Experiment with execution environments, but standardize quickly
- Add enterprise scaling capabilities
 - Will depend on your execution environment/model
 - Build it in the cloud and leverage PaaS options
- Make sure you use CI/CD for long term management

RSA® Conference 2018

San Francisco | April 16 – 20 | Moscone Center



#RSAC

SESSION ID: CSV-R04

PRAGMATIC SECURITY AUTOMATION FOR CLOUD

Rich Mogull

Analyst/VP of Product
Securosis/DisruptOPS
rmogull@disruptops.com
@rmogull

