RSAConference2016
San Francisco | February 29 – March 4 | Moscone Center

Connect to Protect

SESSION ID: CSV-T10

Aspirin as a Service: Using the Cloud to Cure Security Headaches

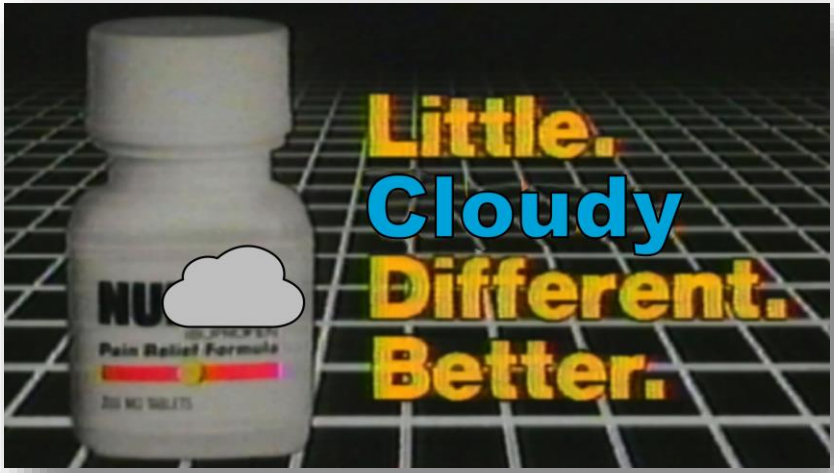
Bill Shinn
Principle Security Solutions Architect
Amazon Web Services

Rich Mogull
CEO
Securosis
@rmogull

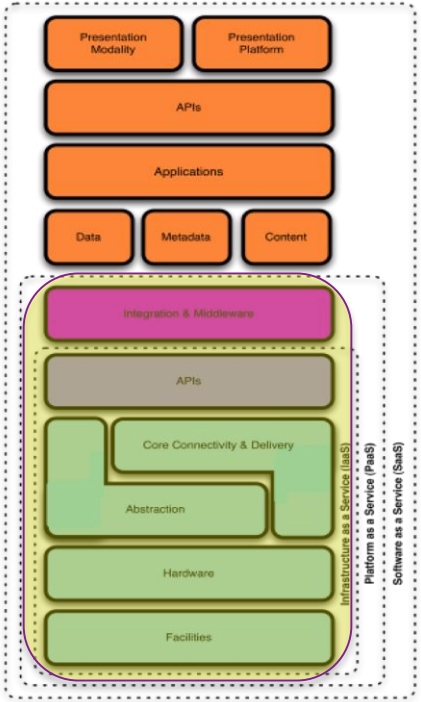
#RSAC

Little. Cloudy. Different.

- Cloud can be more secure than traditional datacenters.
  - The economics are in your favor.
  - Cloud architectures can wipe out some traditional security headaches.
- This isn't theory, it's being done today.
  - But only if you understand how to leverage the cloud.
- We will show you how.



Not the SaaS you're looking for



This session is all IaaS and PaaS

Cloud Security Economics

- For clients to use a cloud provider, they must trust the provider.
- This is especially true for anything with a sensitive data or process.
- Thus security has to be a top priority for a provider or you won't use them.
- A major breach for a provider that affects multiple customers is an existential event.

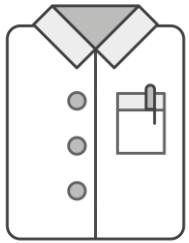
You get one chance

Cloud Provider Critical Security Capabilities

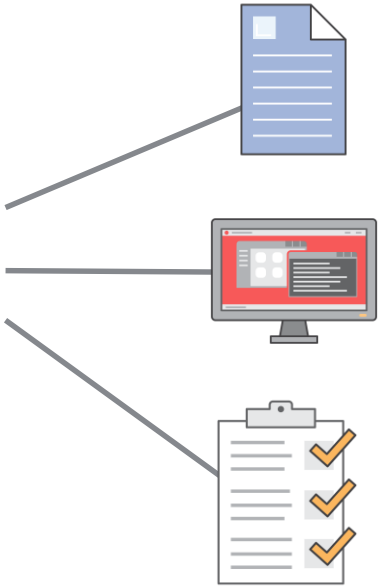
- API/admin activity logging
- Elasticity and autoscaling
- APIs for all security features
- Granular entitlements
- Good SAML support
- Multiple accounts per customer
- Software defined networking
- Region/location control
- *Nice to have: infrastructure templating/automation*

Evolving the Practice of Security Architecture

- Security architecture as a silo'd function can no longer exist.



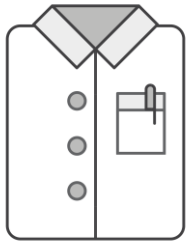
Current Security Architecture Practice



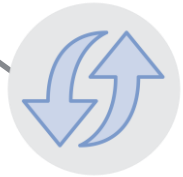
- Static position papers, architecture diagrams & documents
- UI-dependent consoles and "pane of glass" technologies
- Auditing, assurance, and compliance are decoupled, separate processes

Evolving the Practice of Security Architecture

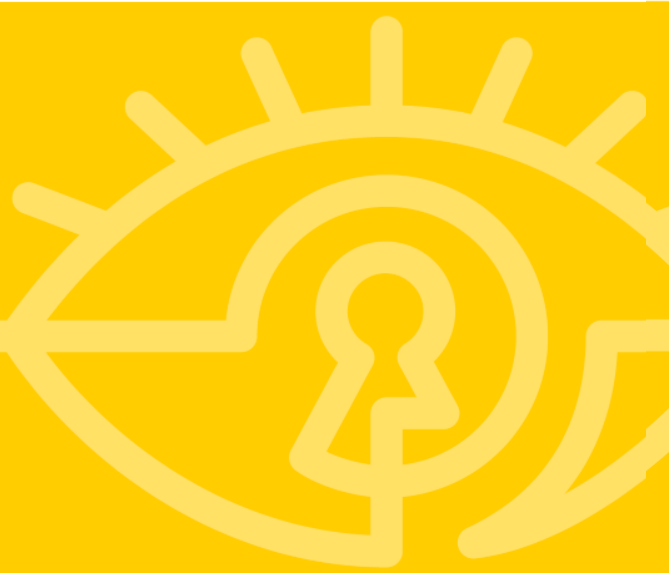
- Security architecture as a silo'd function can no longer exist.



Evolved Security Architecture Practice

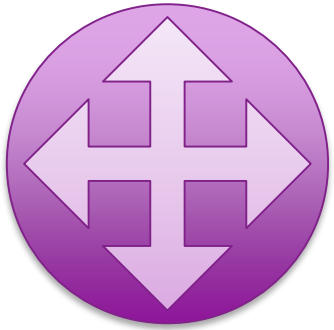


- Architecture artifacts (design choices, narrative, etc.) committed to common repositories
- Complete solutions account for automation
- Solution architectures are living audit/compliance artifacts and evidence in a closed loop



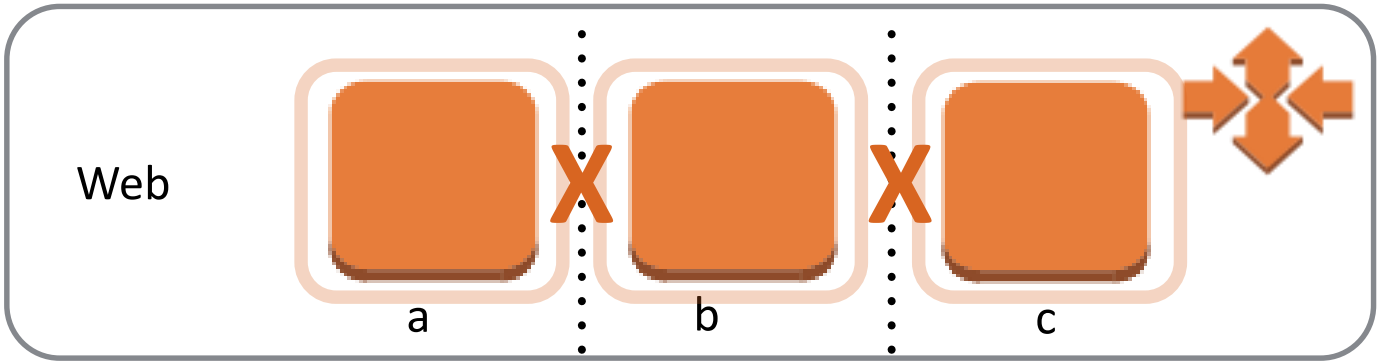
Network Segmentation

Segregation is critical but hard



- Segregating networks in a data center is hard, expensive, and often unwieldy.
- It's hard to isolate application services on physical machines.
  - Even using virtual machines has a lot of management overhead.
- Attackers drop in and move North/South in application stacks, and East/West on networks (or both).

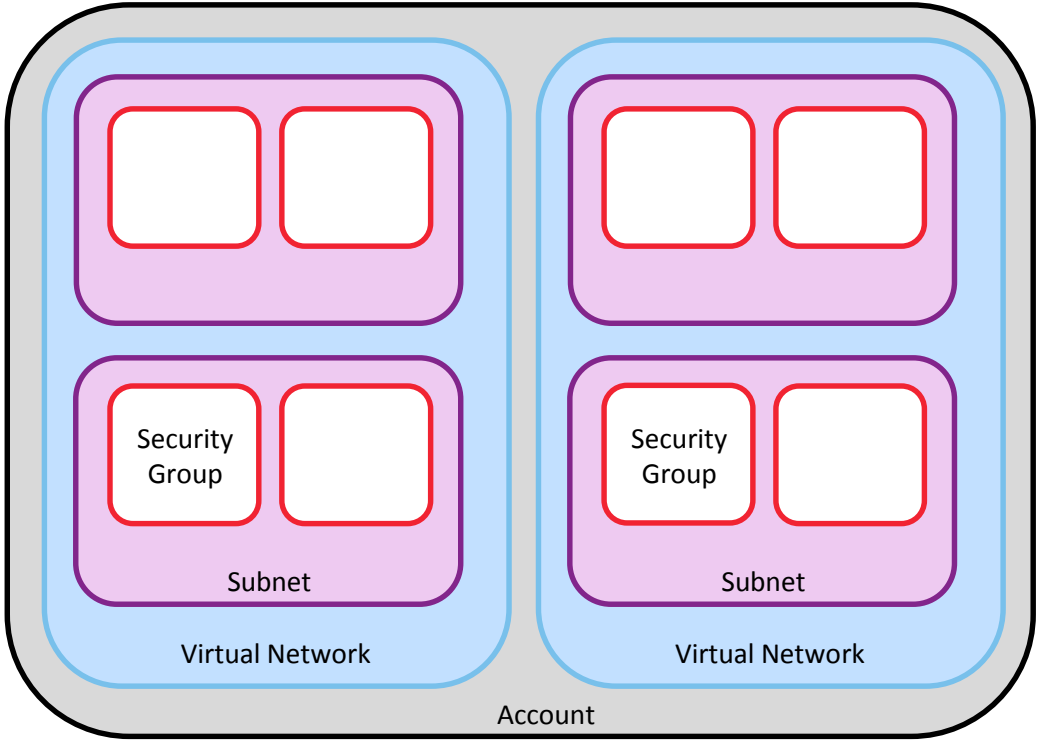
Network segregation by default



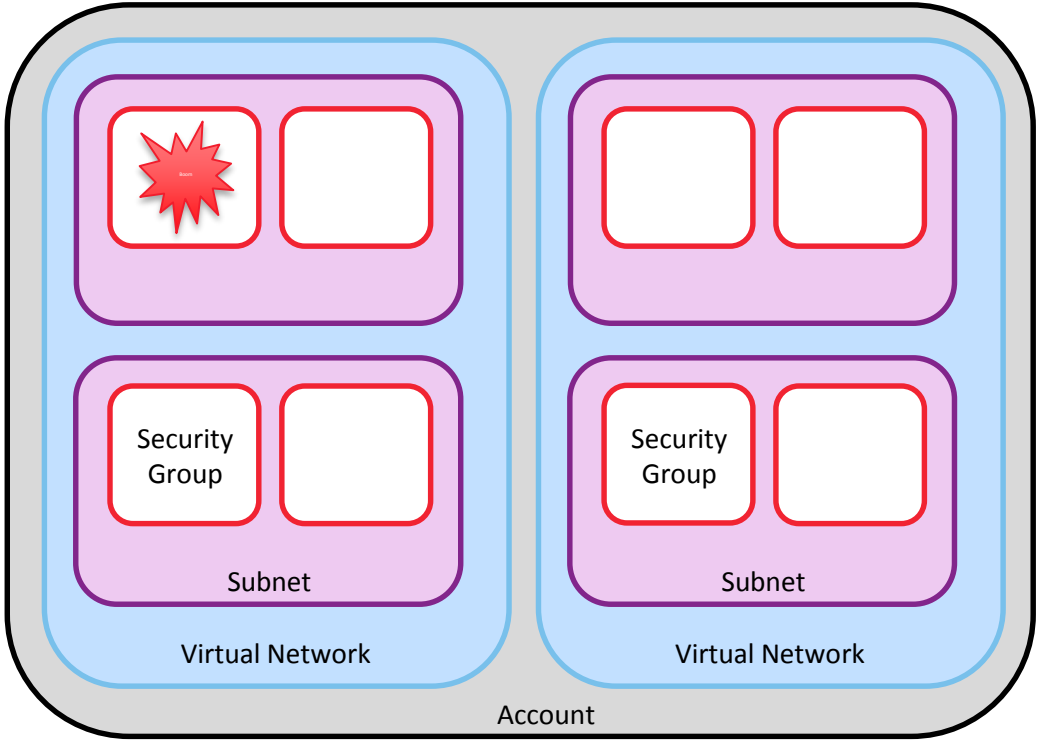
Type	Protocol	Port Range	Source
SSH	TCP	22	68.2.174.98/32
HTTP	TCP	80	0.0.0.0/0

- Granularity of host firewall with ease of management of network firewall

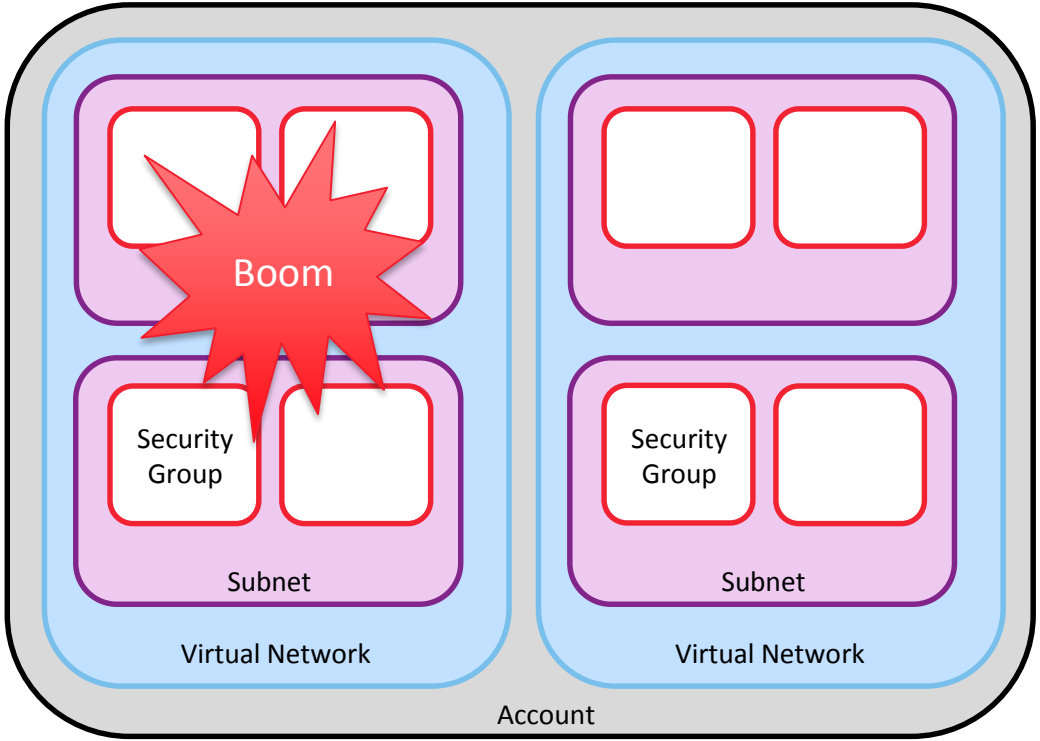
Limiting blast radius



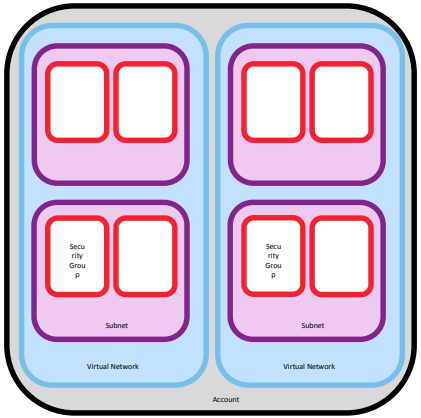
To a host or network…

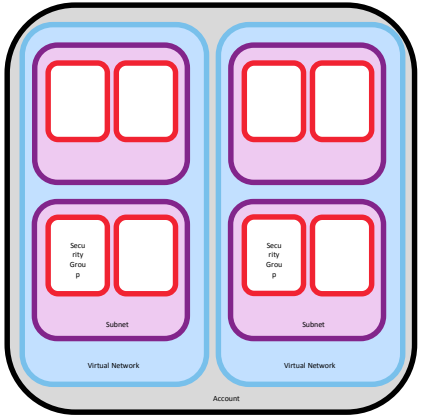


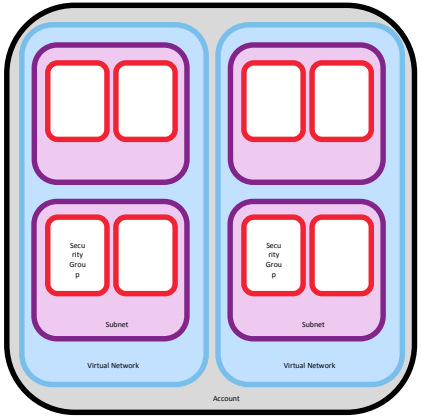
To a host or network…



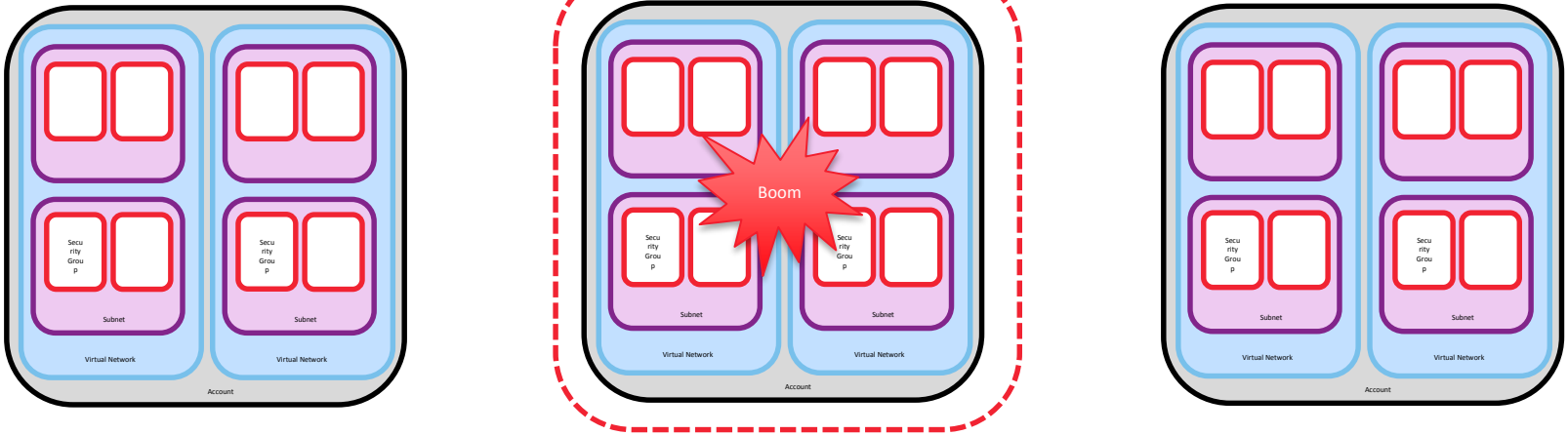
Or an entire “data center”



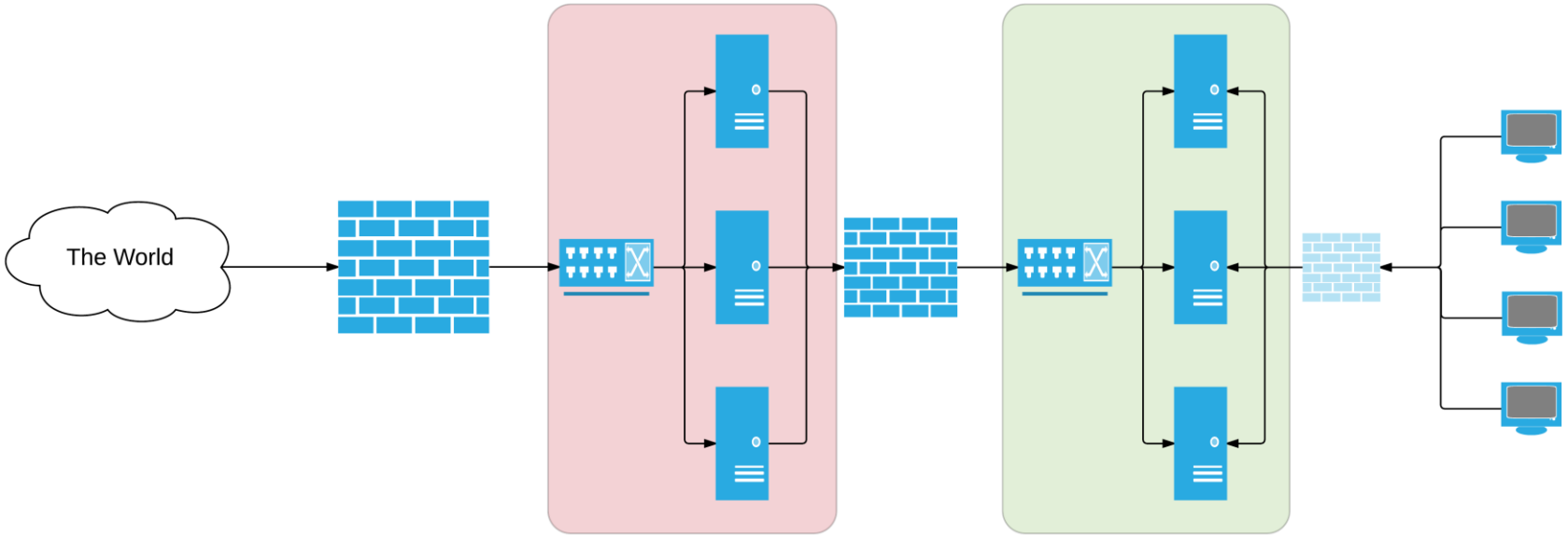




Or an entire “data center”



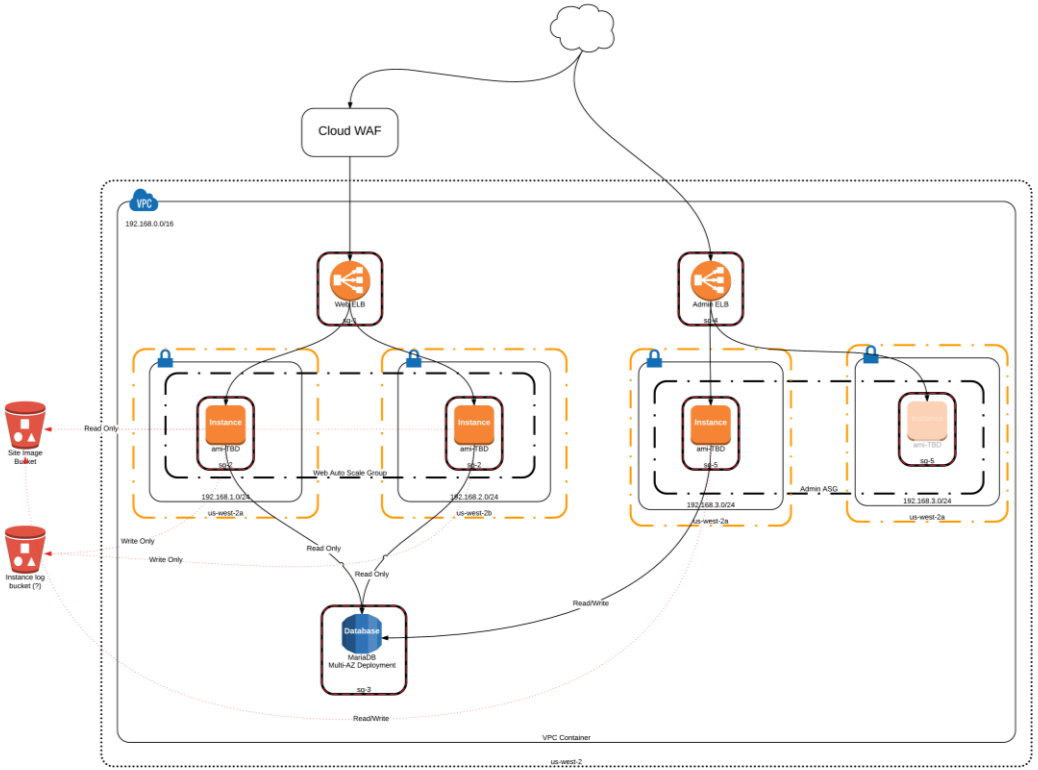
Traditional blast radius



Application segregation

- Easier to deploy smaller services
- Easier to isolate
- Can integrate PaaS for “network air gaps”

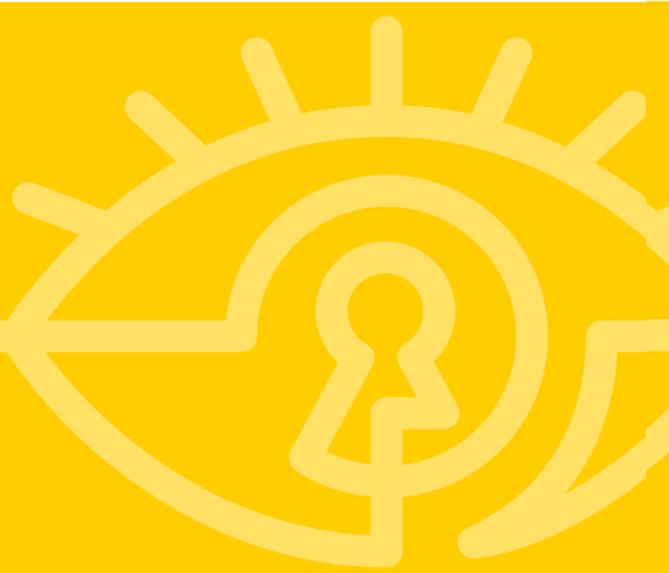
Cloud “DMZ”


Cloud WAF
VPC
192.168.0.0/16
Web ELB
sg-1
Admin ELB
sg-4
Instance
ami-TBD
sg-2
sg-5
Web Auto Scale Group
Admin ASG
192.168.1.0/24
192.168.2.0/24
192.168.3.0/24
us-west-2a
us-west-2b
Site Image Bucket
Instance log bucket (?)
Read Only
Write Only
Read/Write
Database
MariaDB
Multi-AZ Deployment
sg-3
VPC Container
us-west-2

Template

```
{
  "AWSTemplateFormatVersion": "2010-09-09",

    "Parameters" : {
      "KeyName" : {
        "Description" : "The EC2 Key Pair to allow SSH access to the instance",
        "Type" : "String"
      }
    },
  "Resources": {
    "LabVPC": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": "10.0.0.0/16",
        "InstanceTenancy": "default",
        "EnableDnsSupport": "true",
        "EnableDnsHostnames": "true",
        "Tags": [
          {
            "Key": "Name",
            "Value": "CloudSec Lab"
          },
          {
            "Key": "Group",
            "Value": "Vpclab"
          }
        ]
      }
    },
    "subnet667a7420": {
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "CidrBlock": "10.0.0.0/24",
        "AvailabilityZone": "us-west-2c",
        "VpcId": {
          "Ref": "LabVPC"
        },
        "Tags": [
          {
            "Key": "Name",
```

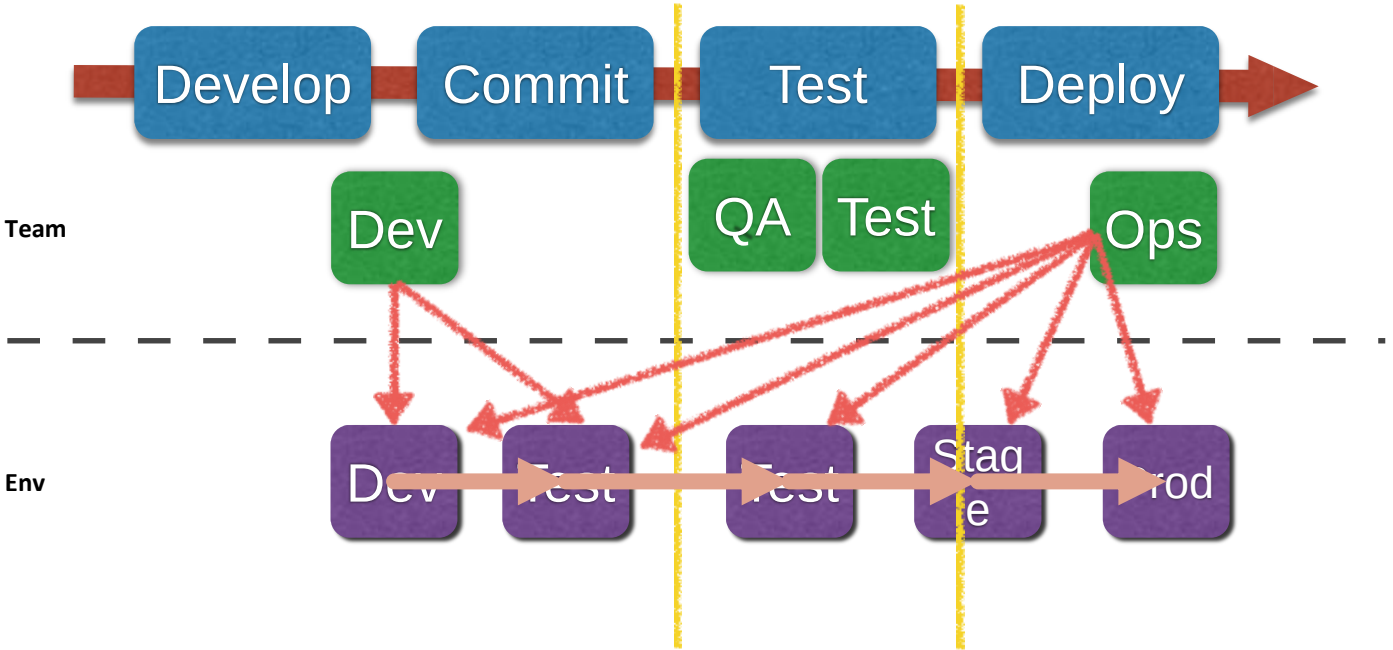


Immutable Services Architectures

Managing patches and change

- Nothing we deploy is consistent.
- Even when we become consistent, it's hard to patch live stuff without breaking things.
- Privileged users log into servers and make changes.
- Attackers love persistent servers they can compromise and camp inside.
- Plus, we need to keep the auditors happy.

Design to deploy is a mess

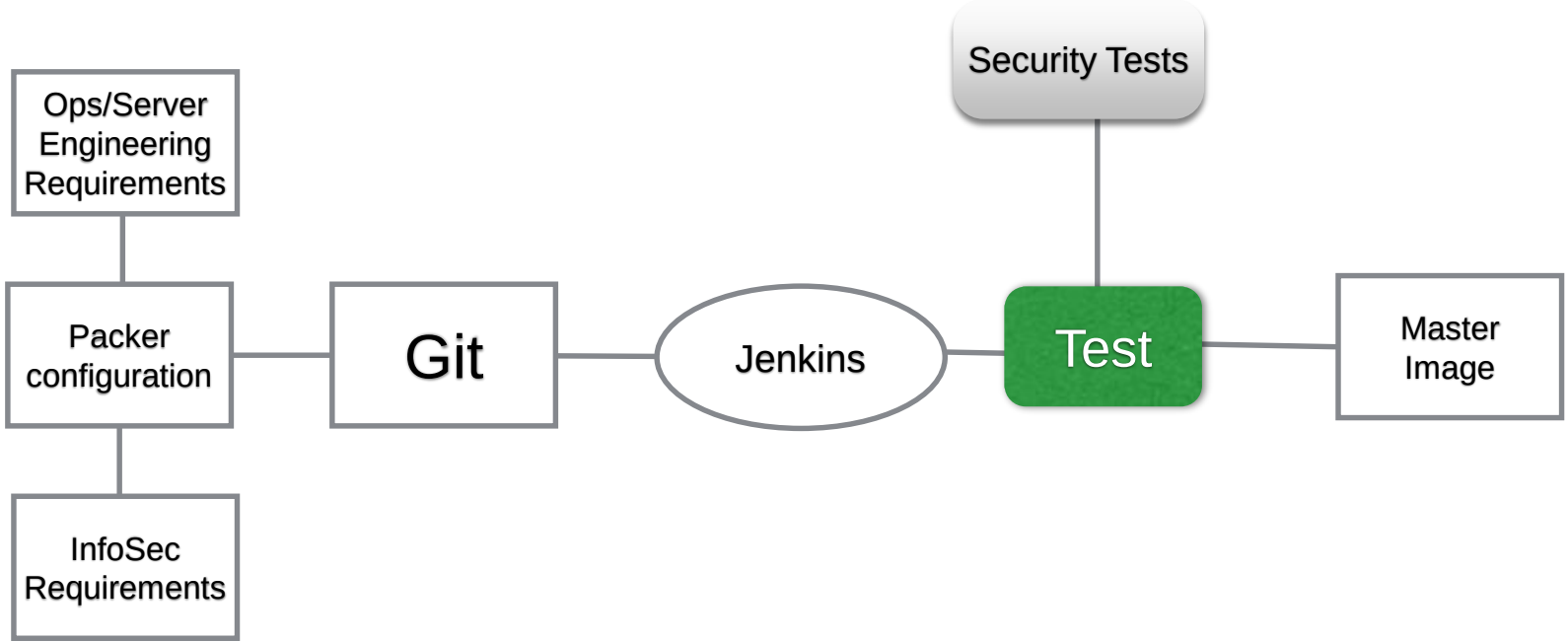


The power of immutable

- Instead of updating, you completely replace infrastructure through automation.
- Can apply to a single server, up to an entire application stack.
- Incredibly resilient and secure. Think “servers without logins”.

Image from: http://tourismplacesworld.blogspot.com/2012/07/uluru.html

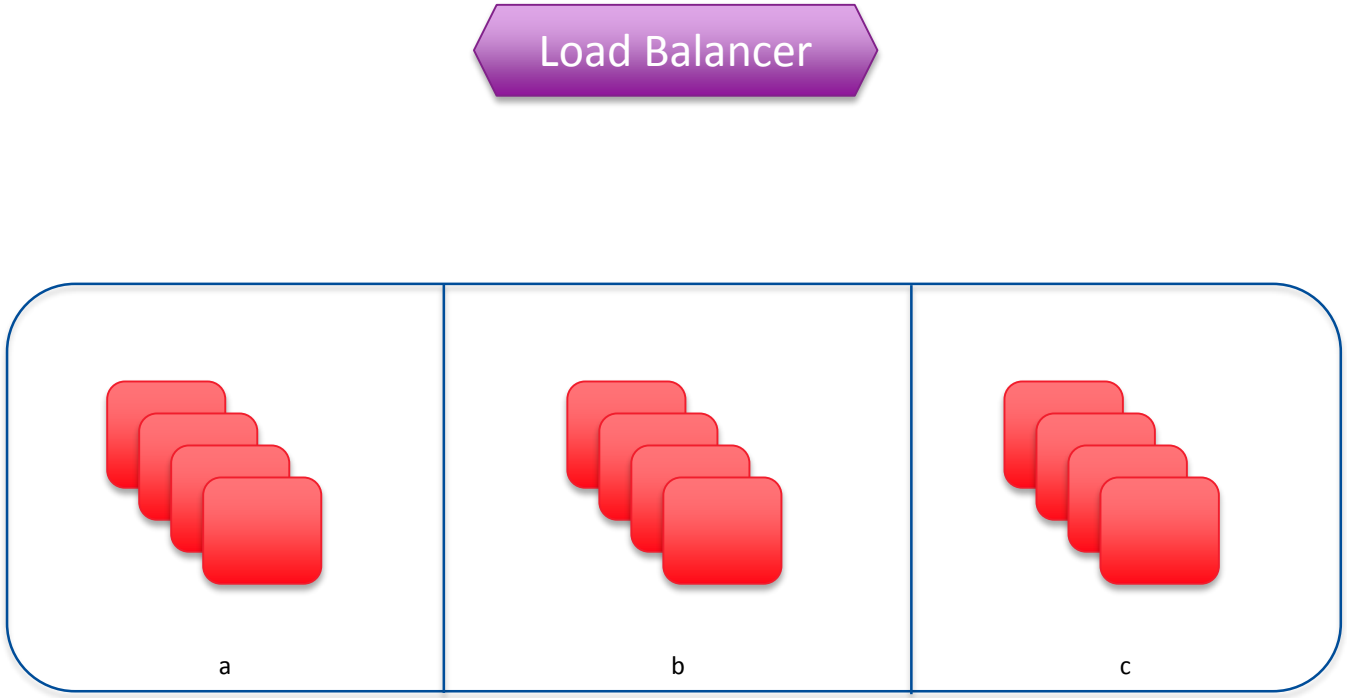
Automate Creation of Master OS Images



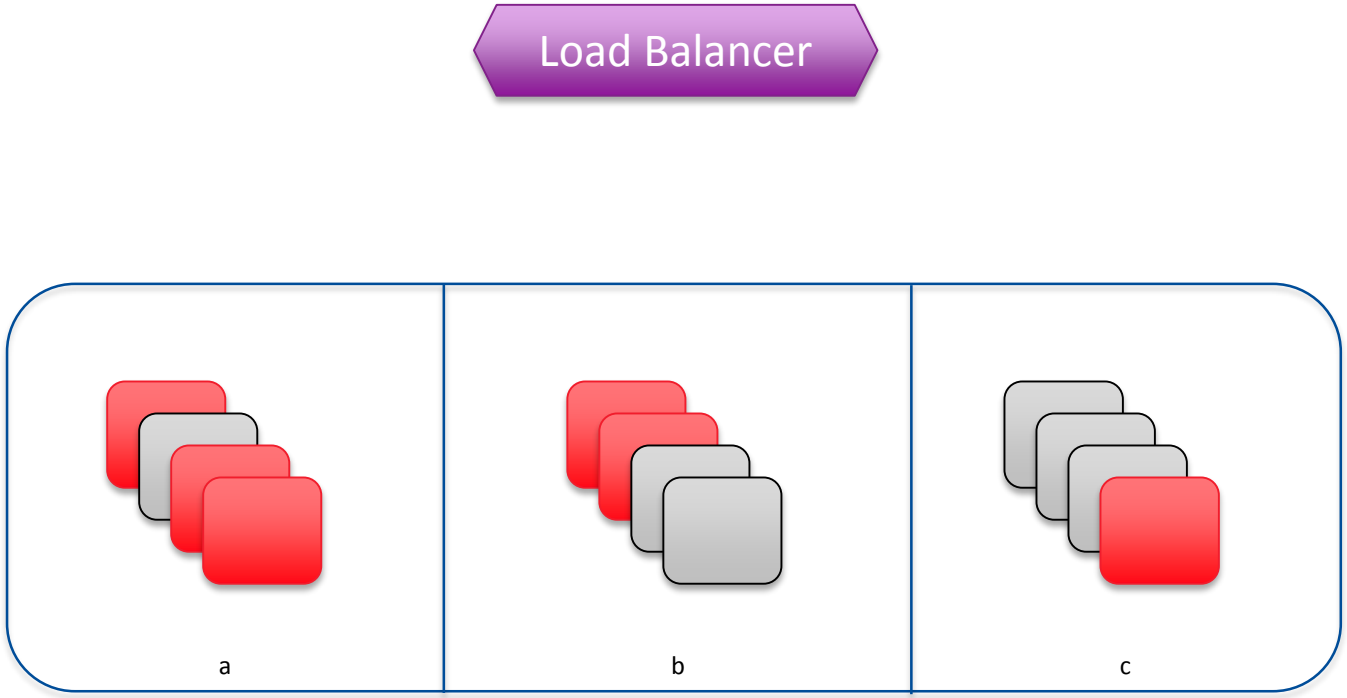
Demo – Server Image Bakery/Factory

- Update the desired configuration of a new master OS image
- Build the master image
- Test the master image for security controls
- Make image available for use

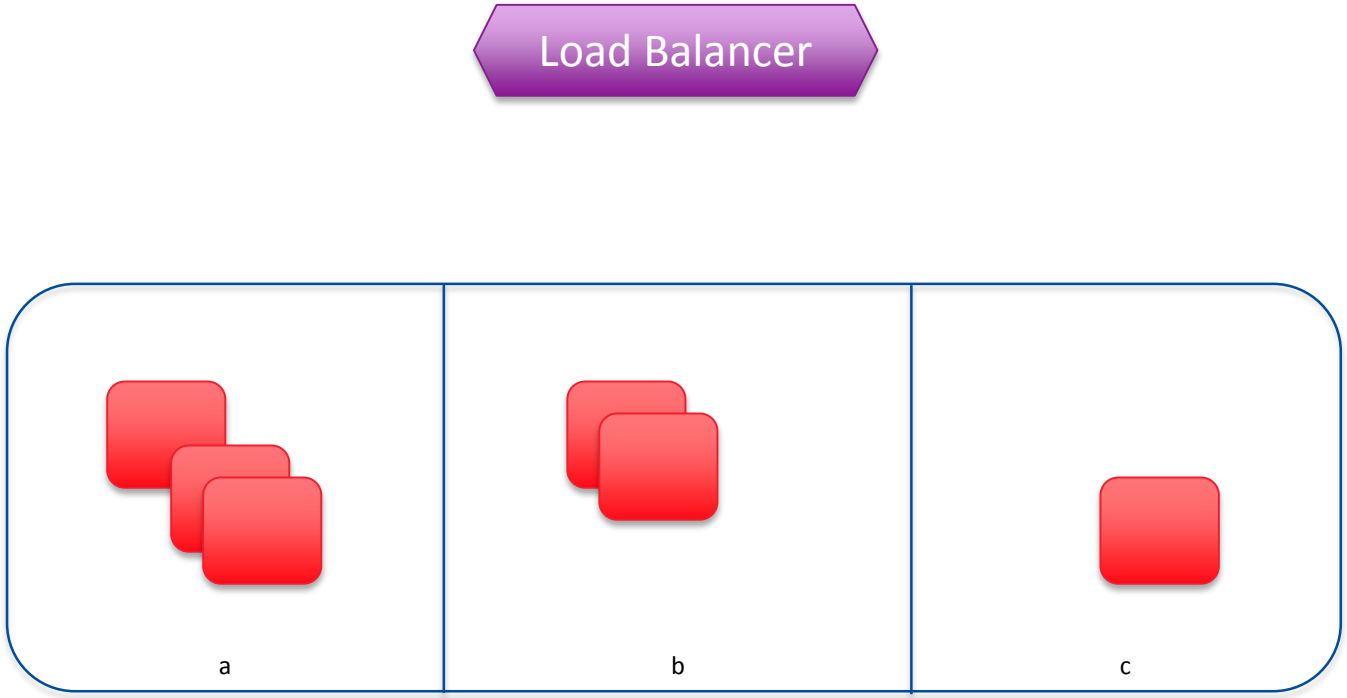
How immutable works- auto scaling



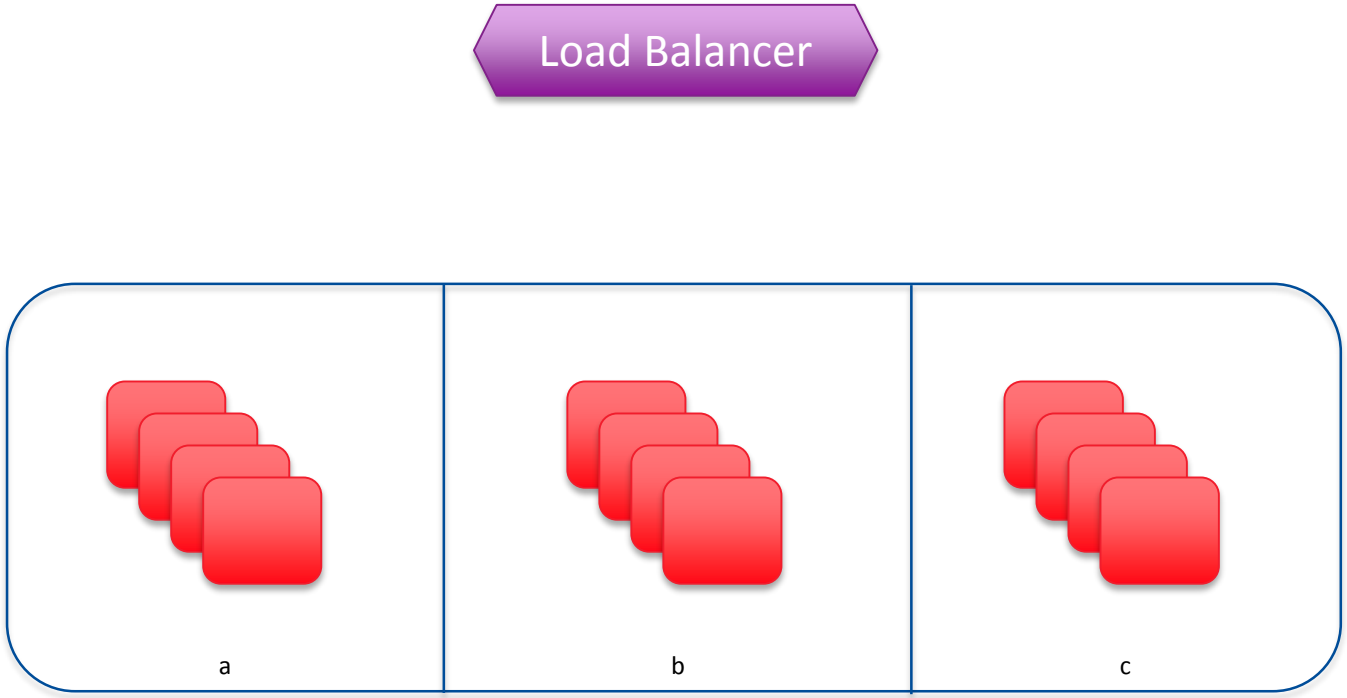
How immutable works- auto scaling



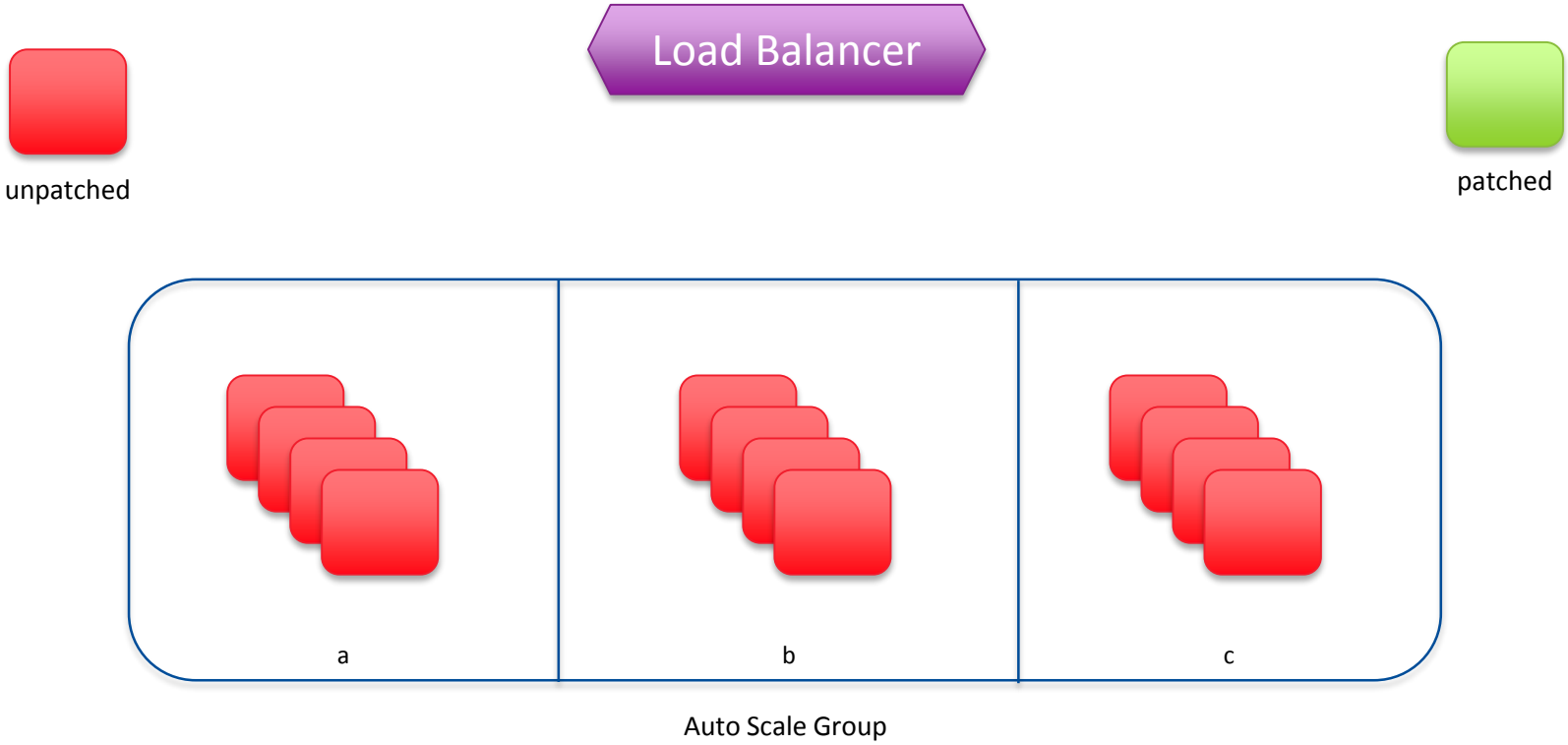
How immutable works- auto scaling



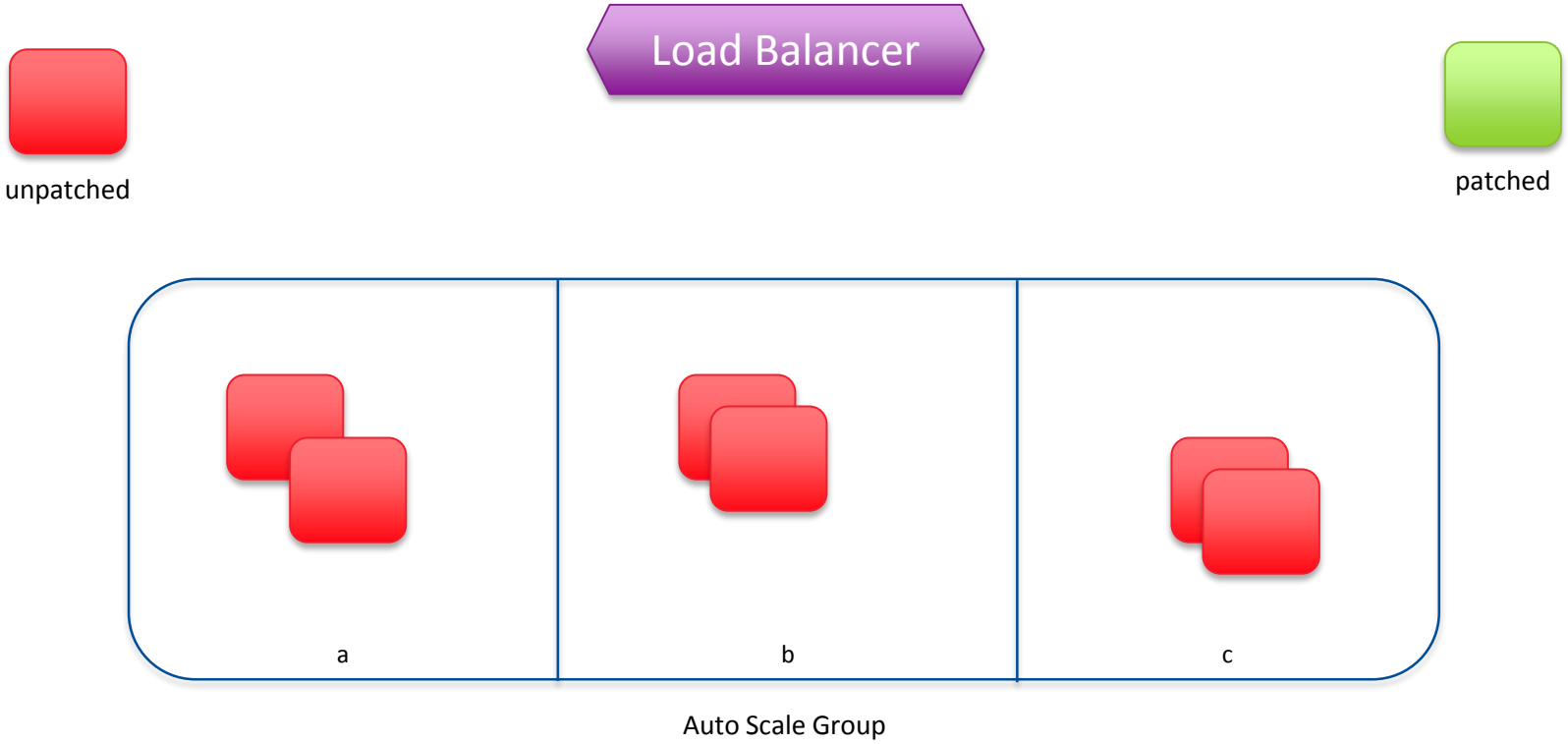
How immutable works- auto scaling



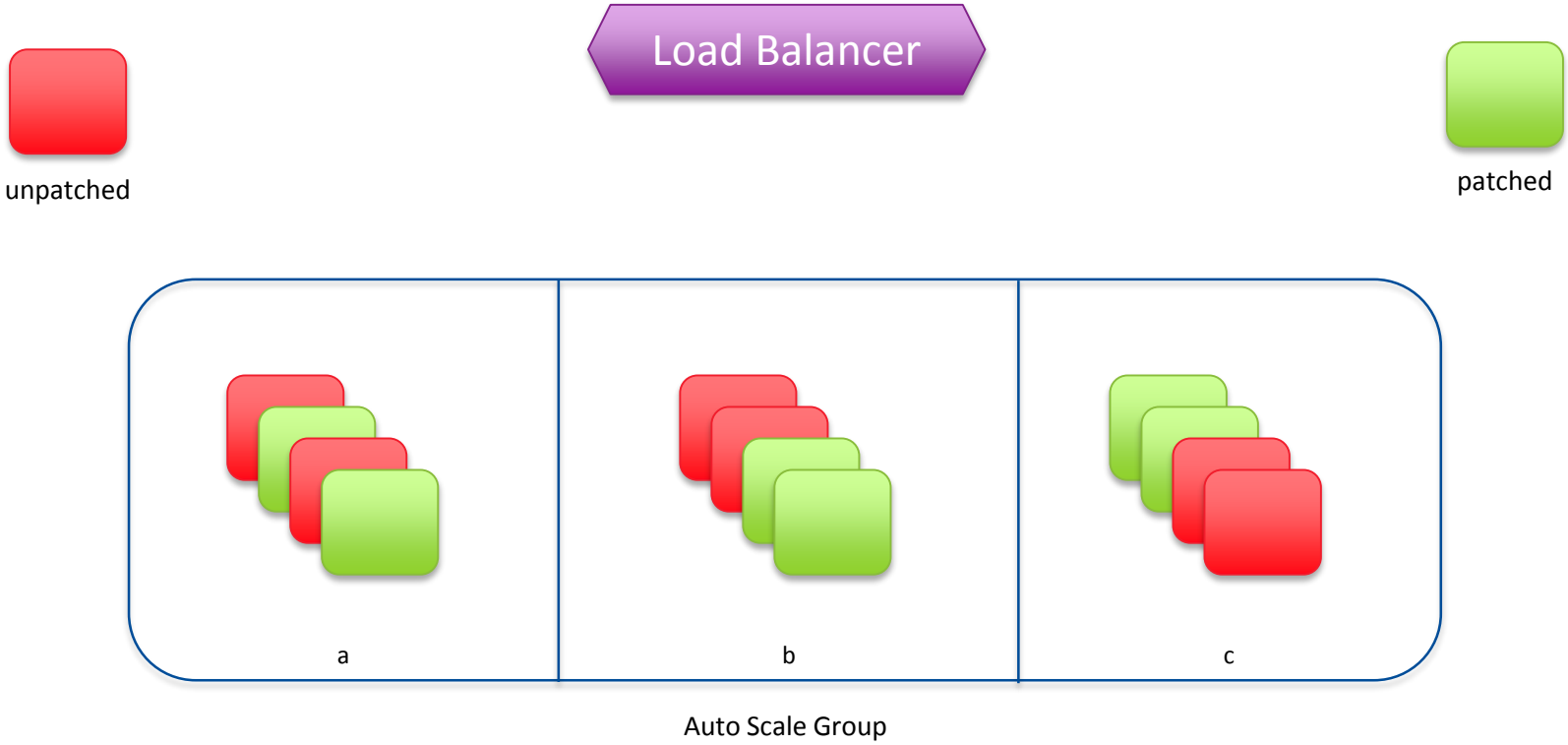
How immutable works- auto scaling



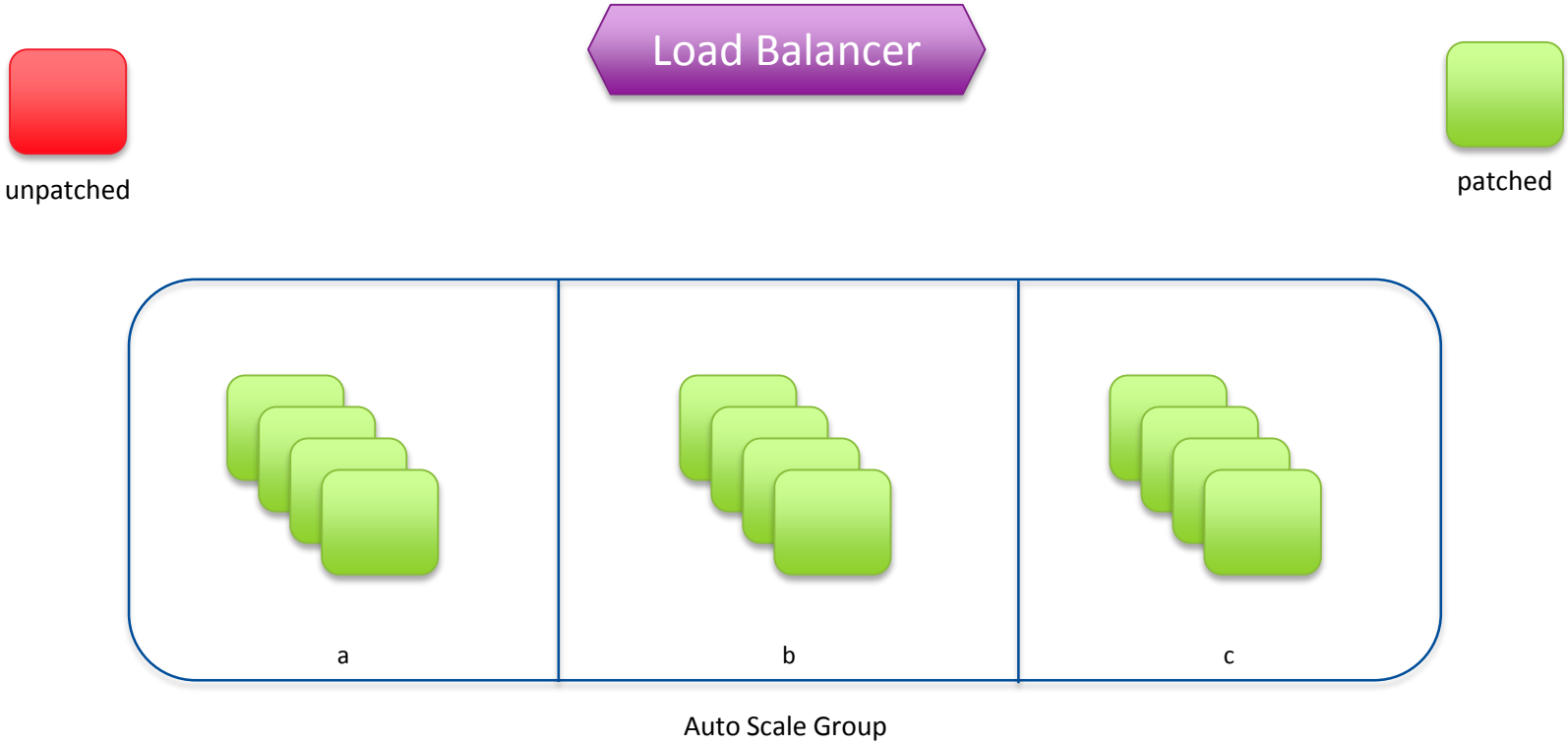
How immutable works- auto scaling



How immutable works- auto scaling

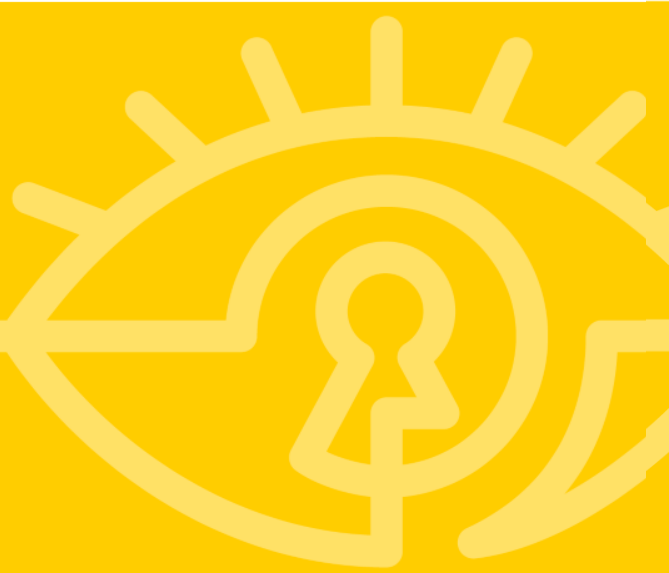


How immutable works- auto scaling



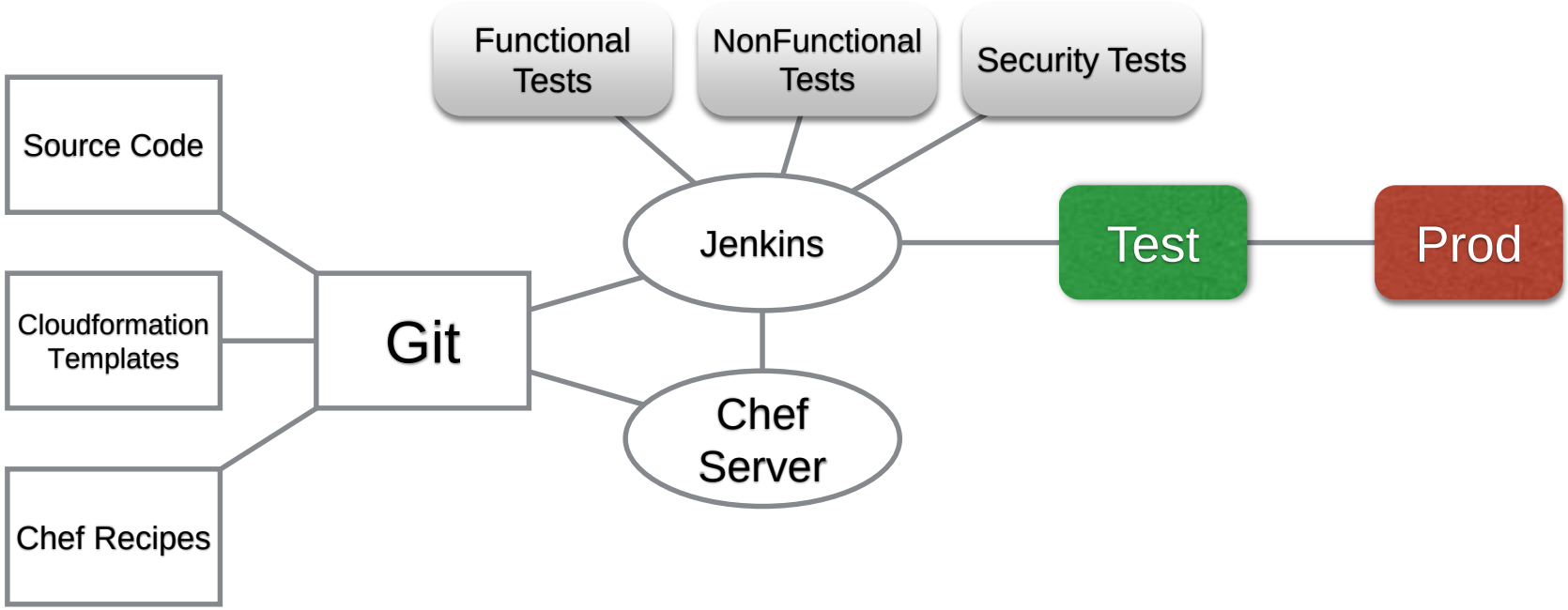
Demo

- Rolling update of 40 instances in 4 minutes with 0 downtime.

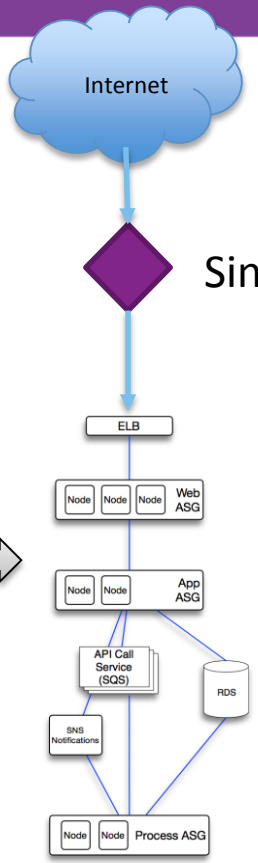


Immutable Infrastructure

Automate with DevOps and Continuous Deployment



Immutable Infrastructure

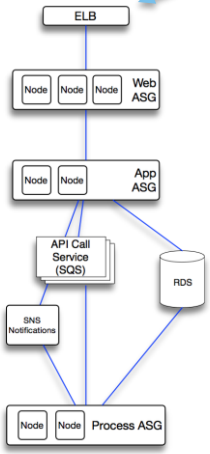


Template A:

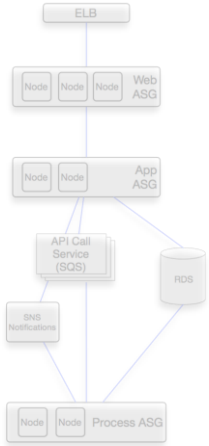
Immutable Infrastructure

Launch updated version

Template A:



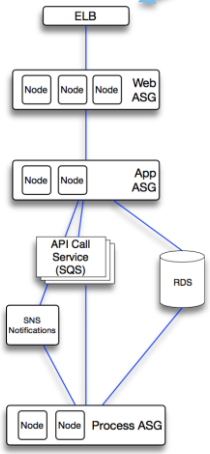
Template B:



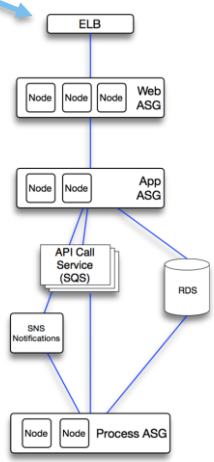
Immutable Infrastructure

Begin diverting traffic via DNS

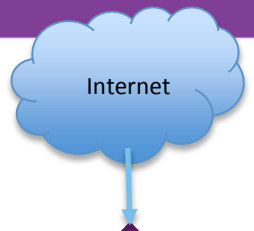
Template A:



Template B:

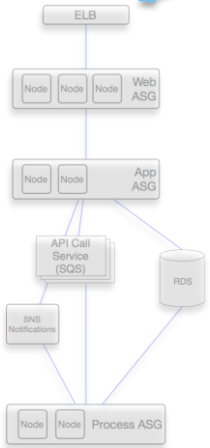


Immutable Infrastructure

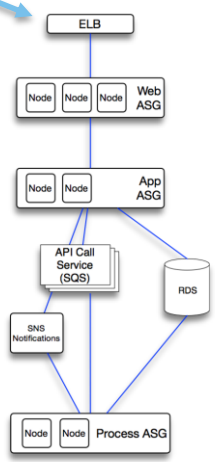


Rollback or finish, depending on results

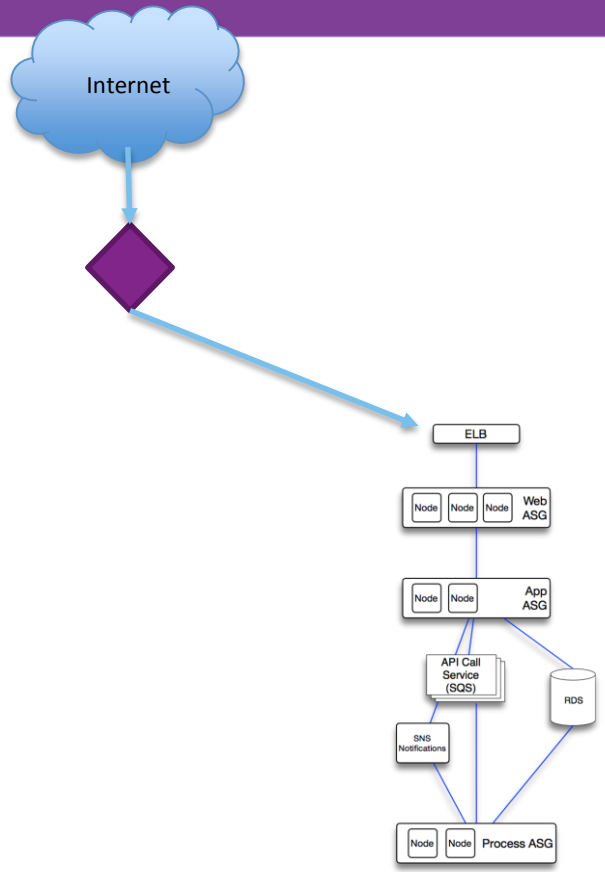
Template A:



Template B:

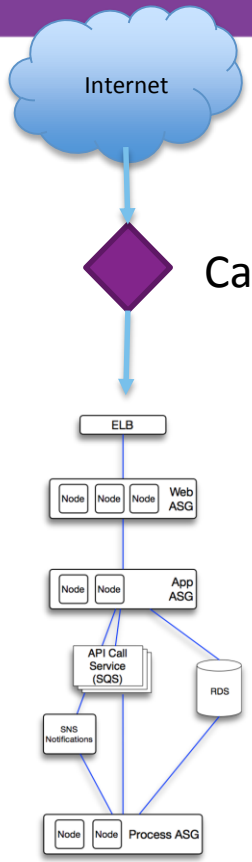


Immutable Infrastructure



Template B:

Immutable Infrastructure



Can *still* roll back if needed

Template B:

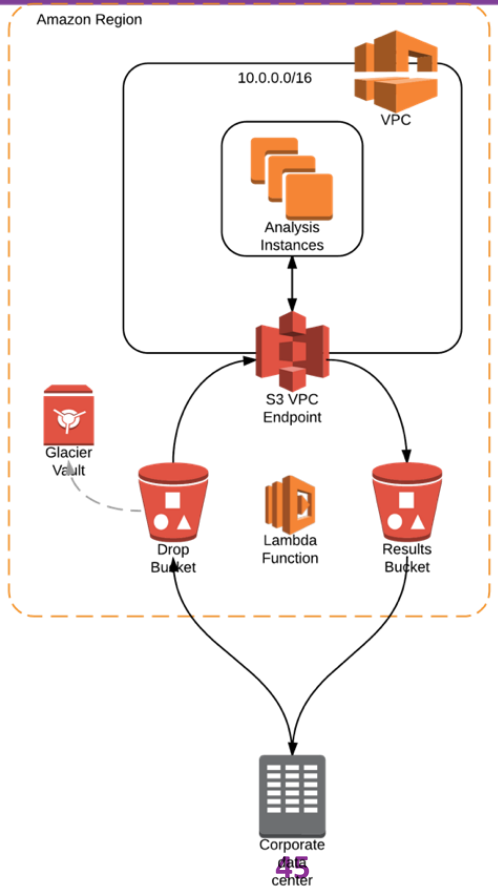
Let your PaaS do the work

- We deploy many MANY core components to deliver applications.
  - Load balancers, databases, message queues, and more.
- It takes a lot of effort to keep these secure and up to date at scale.
- Each piece is yet more attack surface.

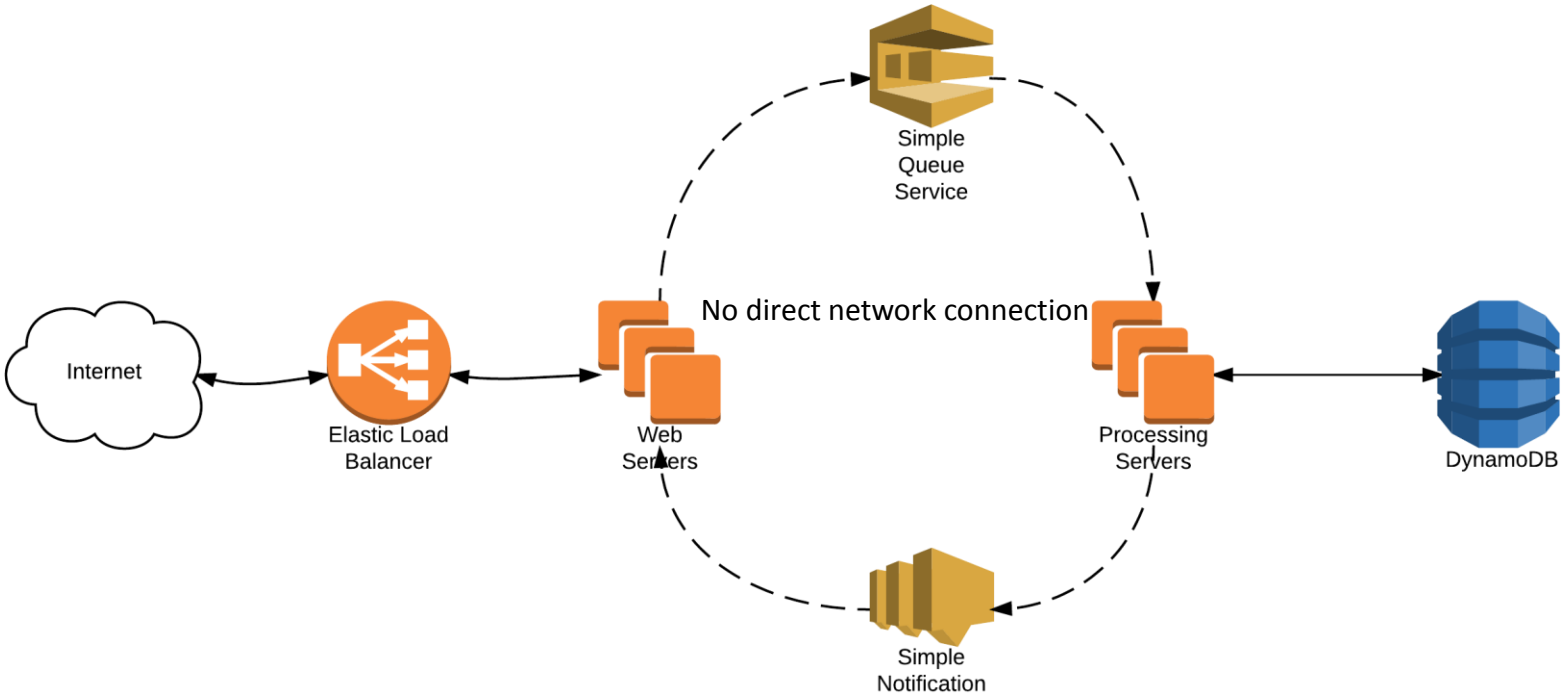
PaaS and ”New” Cloud Architectures

- PaaS providers can’t afford a preventable security failure.
  - Including letting things get out of date.
- Many types of PaaS can’t rely on normal networking.
  - Instead you access them via API.
- This creates an opportunity to “air gap” parts of your application.
  - Kill off network attack paths (doesn’t help with logic flaws)

Network attack path?



PaaS Air Gap

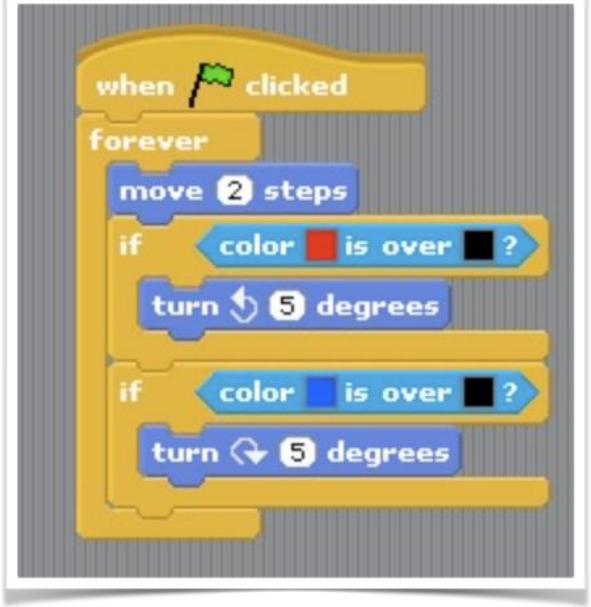

Simple
Queue
Service
No direct network connection
Internet
Elastic Load
Balancer
Web
Servers
Processing
Servers
DynamoDB
Simple
Notification

Software Defined Security

- Attackers are automated, we are mostly manual.
- Our tools have been poor.
- We lack trustable security automation and thus need to rely on a “Meat Cloud”
- In cloud, APIs are mandatory. We can write code to automate and orchestrate, even across products and services.

Code without Coding

- Work with your devs to build a library of building blocks
- Learn just enough to glue it together
- Build some core scripts
- Mix and match the blocks
- Pull in the dev when you have new requirements



Demo

- Meet SecuritySquirrel, the first warrior in the Rodent Army (apologies to Netflix).
- The following tools are written by an analyst with a Ruby-for-Dummies book.
- Automated security workflows spanning products and services.

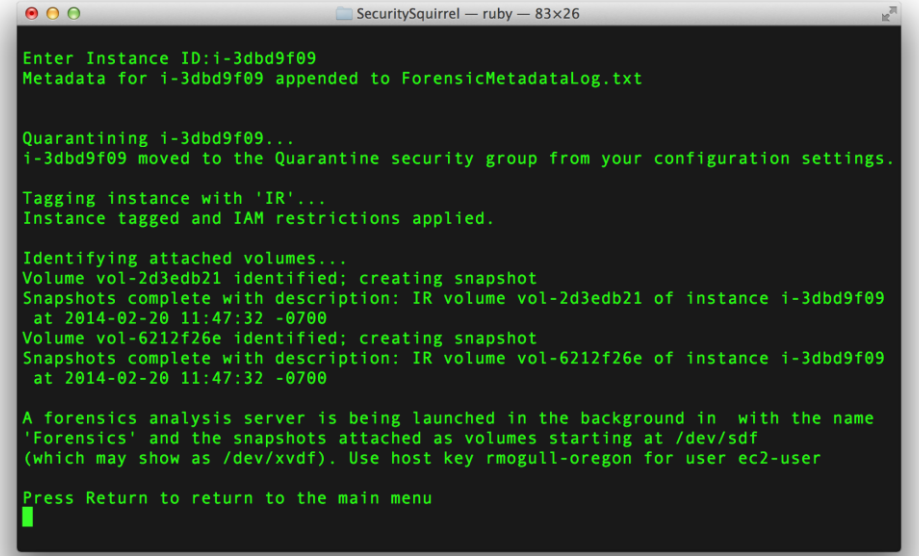
Incident Response

1. Detect Compromise
2. Pull server information (If you have it)
3. Quarantine
4. Image 
5. Analyze
6. Recover

= Hours!

Each step is manual, and uses a different set of disconnected tools

DEMO



1. Pull metadata
2. Quarantine
3. Swap control to security team
4. Identify and image all storage
5. Launch and configure analysis server
6. Can re-launch clean server instantly

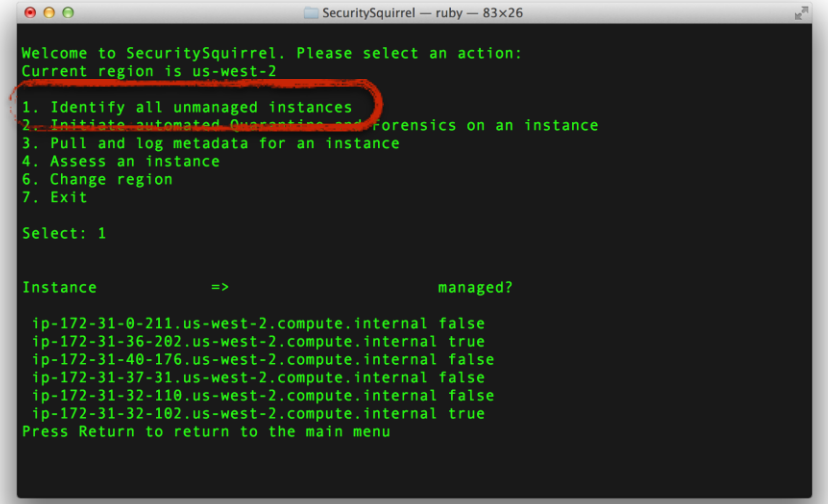
Stateless Security

- Security normally relies on scanning and checking databases.
- With cloud we are completely integrated into the infrastructure and platforms.
  - The cloud controllers have to see everything to manage everything, there is no Neo running around.
- Instead of scanning, we can directly pull state.
  - And then use it for security

Identify Unmanaged Servers (for the audit)

1. Scan the network
2. Scan again and again for all the parts you missed
3. Identify all the servers as best you can
4. Pull a config mgmt report
5. Manually compare results

DEMO

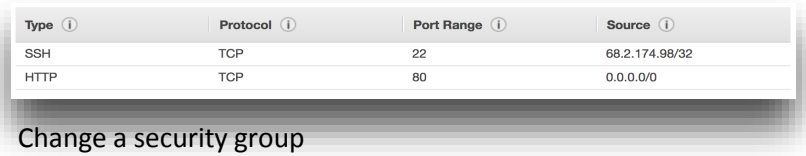


1. Get list of all servers from cloud controller (can filter on tags/OS/etc).
   - Single API call
2. Get list of all servers from Chef
   - Single API call
3. Compare in code

Event Driven Security

- Cloud providers are creating hooks to trigger actions based on events inside the cloud.
- We can use these for near-instant security reactions.

Self-Healing Infrastructure (yes, for real)



Change a security group

Lambda Function analyzes and reverses

Event Recorded to CloudTrail

Passed to CloudWatch Log Stream

Triggers an CloudWatch Event

Demo

- Watch a security group self heal in less than 10 seconds…

Aspirin Applied

- Next week you should:
  - Follow up this session by learning to use Git (or another source repo) and a build pipeline toolchain like Jenkins.
- In the first three months following this presentation you should:
  - Be collaborating with dev/engineering/operations/security on something - anything! Even if you just keep basic “account governance” scripts in a repo that people can run, contribute to, track, build into pipelines, etc- have at least one key security capability wired up through a pipeline.
- Within six months you should:
  - Be running audits out of the toolchain for at least a few key controls as they are applied to the cloud.