RSAConference2019
San Francisco | March 4–8 | Moscone Center

SESSION ID: CSV-W03

Securing Cloud-Native Applications at Scale

Ashwin Ambekar

Principal Security Architect
eBay Inc
@ashwin_ambekar

What is a Cloud Native Application?

Applications that are designed to run ***natively*** on the Cloud. Such Applications are ***elastic***, ***resilient***, ***loosely coupled*** to infrastructure, use ***distributed data*** and can be secure even in public Cloud

Cloud Native Application adds value to the business in a **flexible**, **continuous**, **highly available** and **secure** manner

Cloud-Native Transformation

- Monoliths broken down into multiple functional and non-functional micro-services
- Common abstractions across services stack (e.g. discovery, rate limiting etc.)
- Ubiquitous deployments, independent of environment and location
- Short development cycles and frequent deployments
- Partitioned, duplicated and distributed data

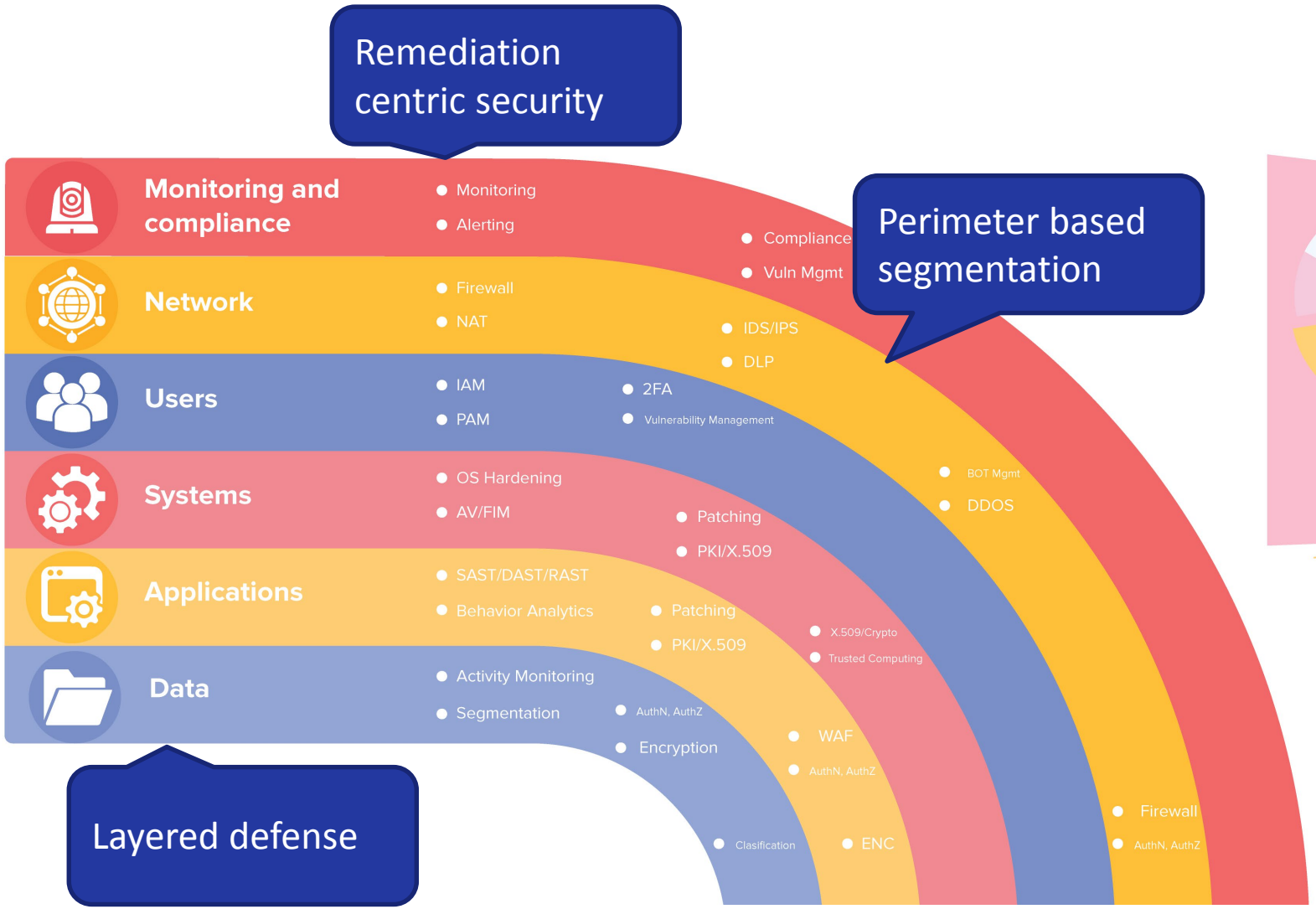
Cloud-Native Transformation at Scale

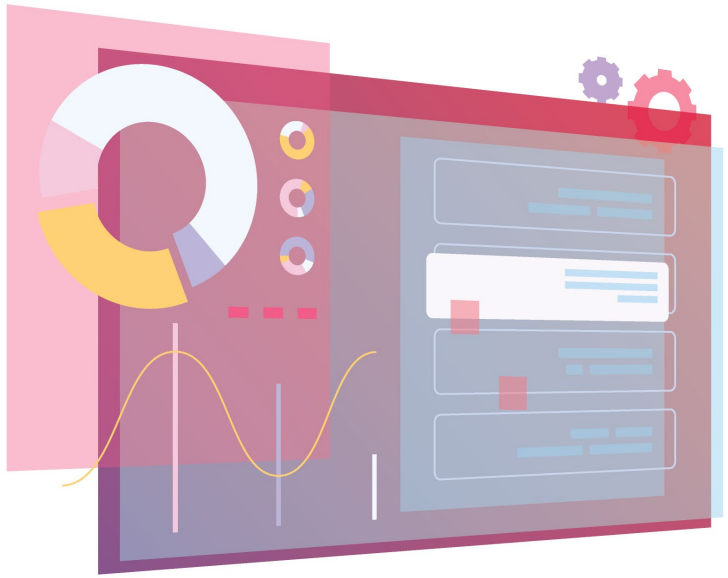
- eBay 2017 Cloud landscape: Before Cloud-Native transition
  - 200,000+ Computes/VMs
  - 1.1+ M deployments
  - 4000+ Applications
- eBay 2019 Cloud landscape: During Cloud-Native transition
  - 336,000+ Computes/VMs
  - 2+ M deployments
  - 20,000+ Applications

Security Implications

- Increased surface area
  - Computes, dependencies, locations
- Dynamic eco-system
  - Ephemeral computes and frequent deployments
- Distributed data and services
- New technology landscape and culture

Traditional Security Model

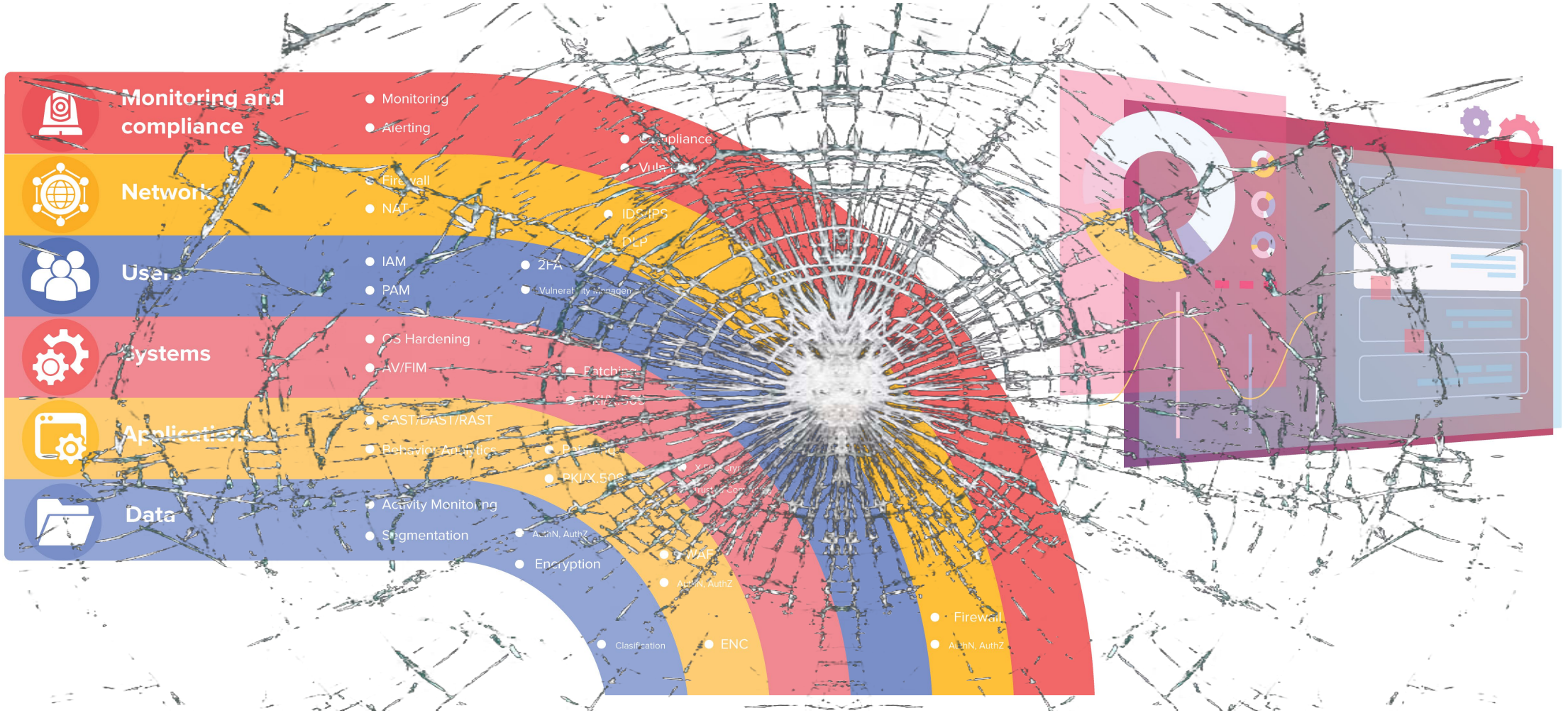




Gaps and challenges

Challenge	Caused by
Observability, Compliance	Increased Surface Area, Ephemeral computes
Scanning, Patching, AV, FIM	Increased Surface Area, Ephemeral computes
PAM, MAC, Access Control	Increased Surface Area, New technology landscape
Data Classification challenges	Distributed Data
X.509	Ephemeral computes, Dynamic Services
DDoS, IDS/IPS, DLP	Encrypted traffic
Segmentation (Network, Apps)	Increased Surface Area, Ephemeral computes
Software Supply Chain	Increased Surface Area

Security model is broken in Cloud-Native Era



Target State

- Layered security controls: Defense in depth
- Application centric security
- Reduce remediation centric security
  - Does not scale

Pillars of Cloud-Native Security at scale

Declarative Security

- Cloud native systems are declarative and intent driven, security is no different

Self Healing

- Drift between declared and actual can be eliminated by self-healing systems
- Self healing systems embrace visibility as basic requirement

Cloud-Native Security

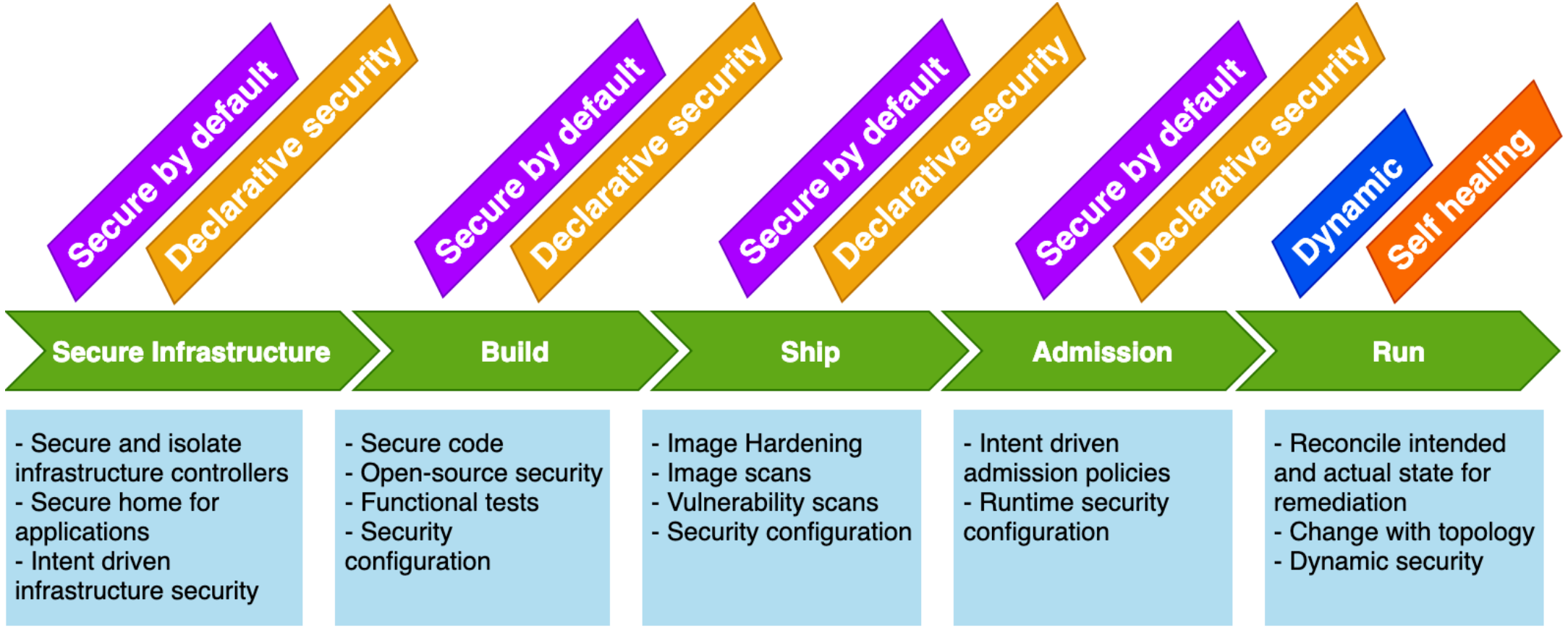
Secure by default

- Embrace immutable infrastructure
- Implement Secure by default policy for code and infrastructure to reduce security gaps

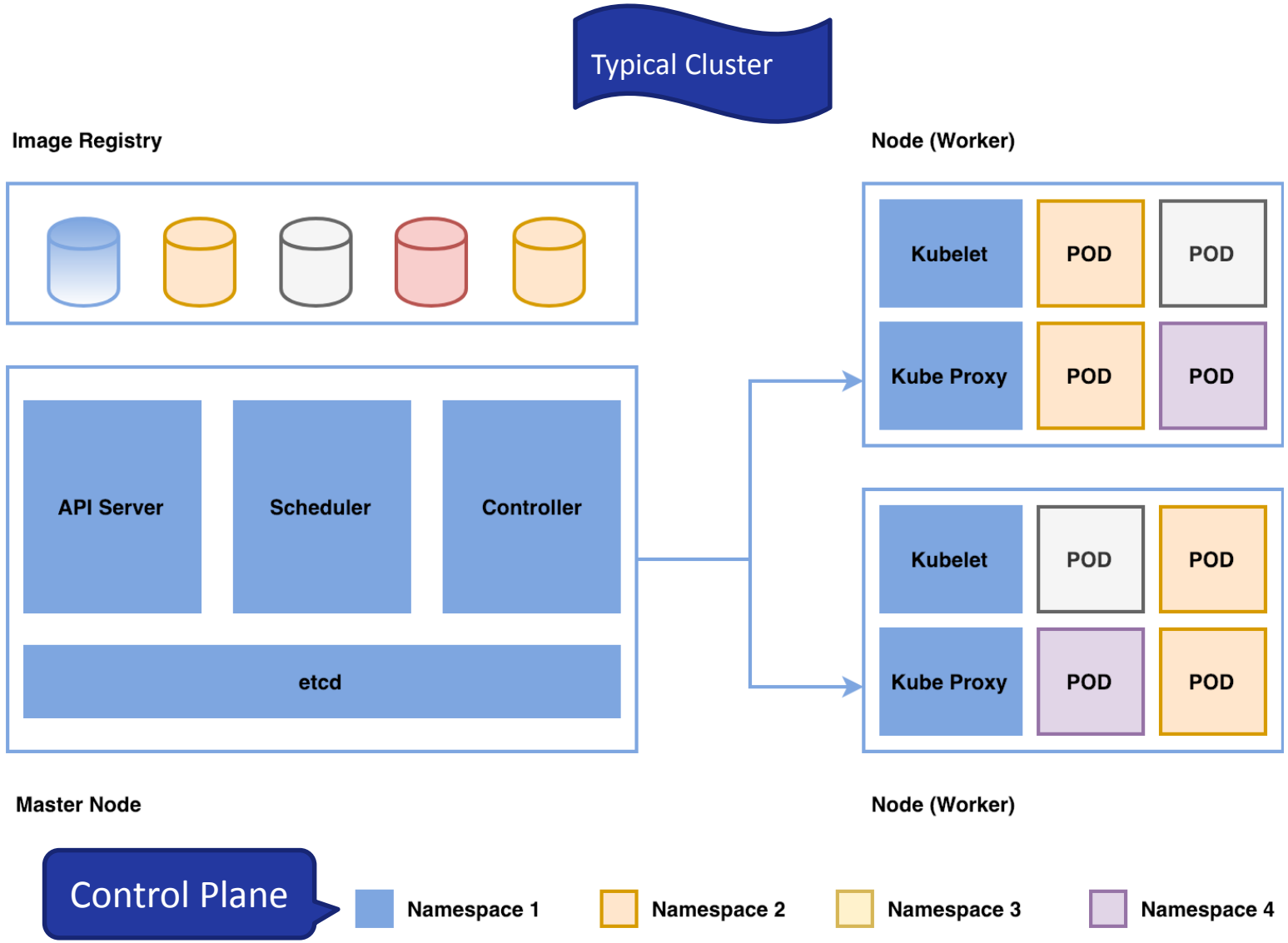
Dynamic

- Process, Controls and Policies must be dynamic and continuous in nature

Securing DevOps

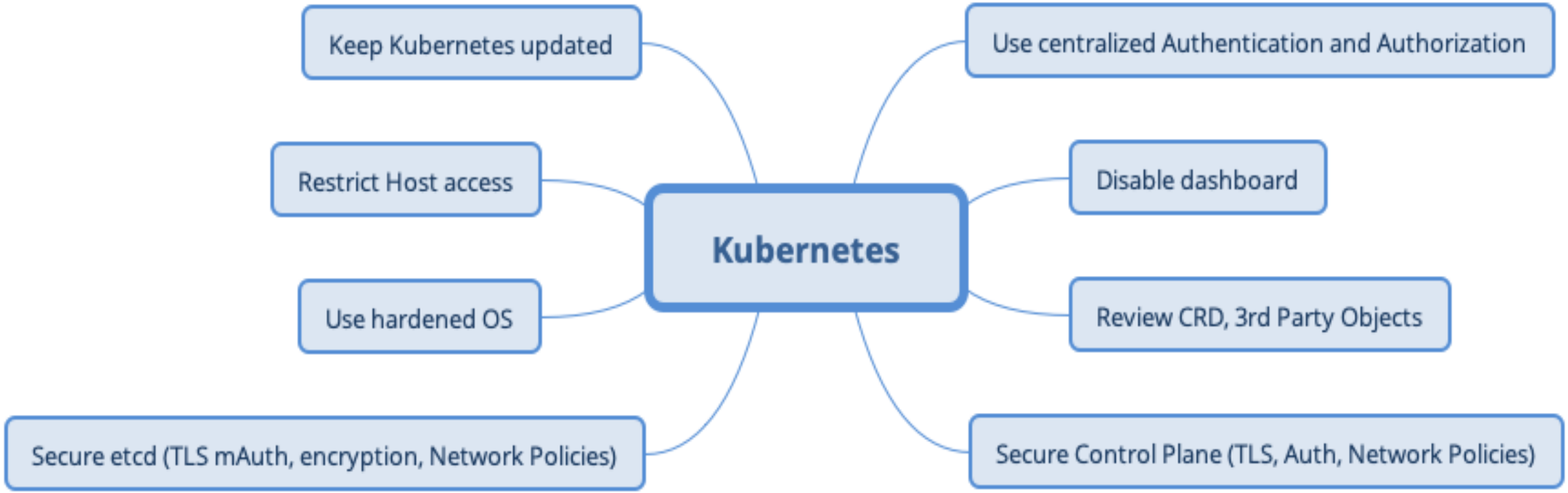


Reference Architecture: Kubernetes

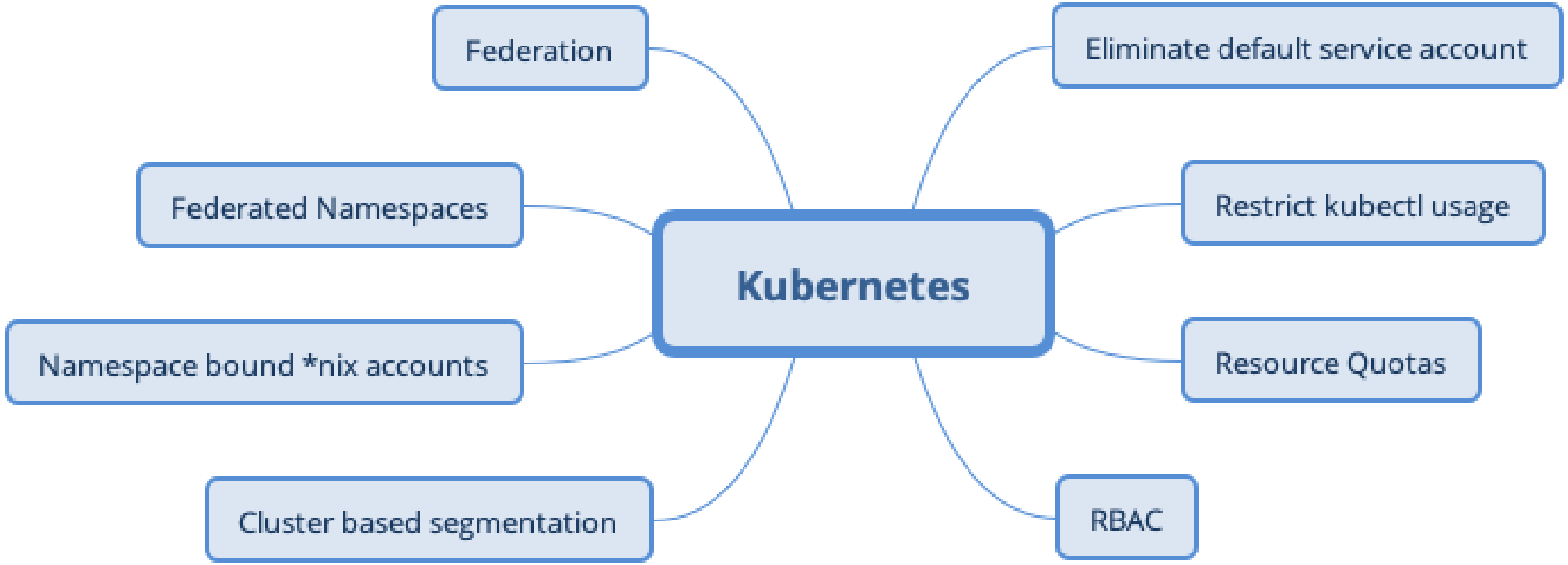


Secure Infrastructure | Build | Ship | Admission | Run

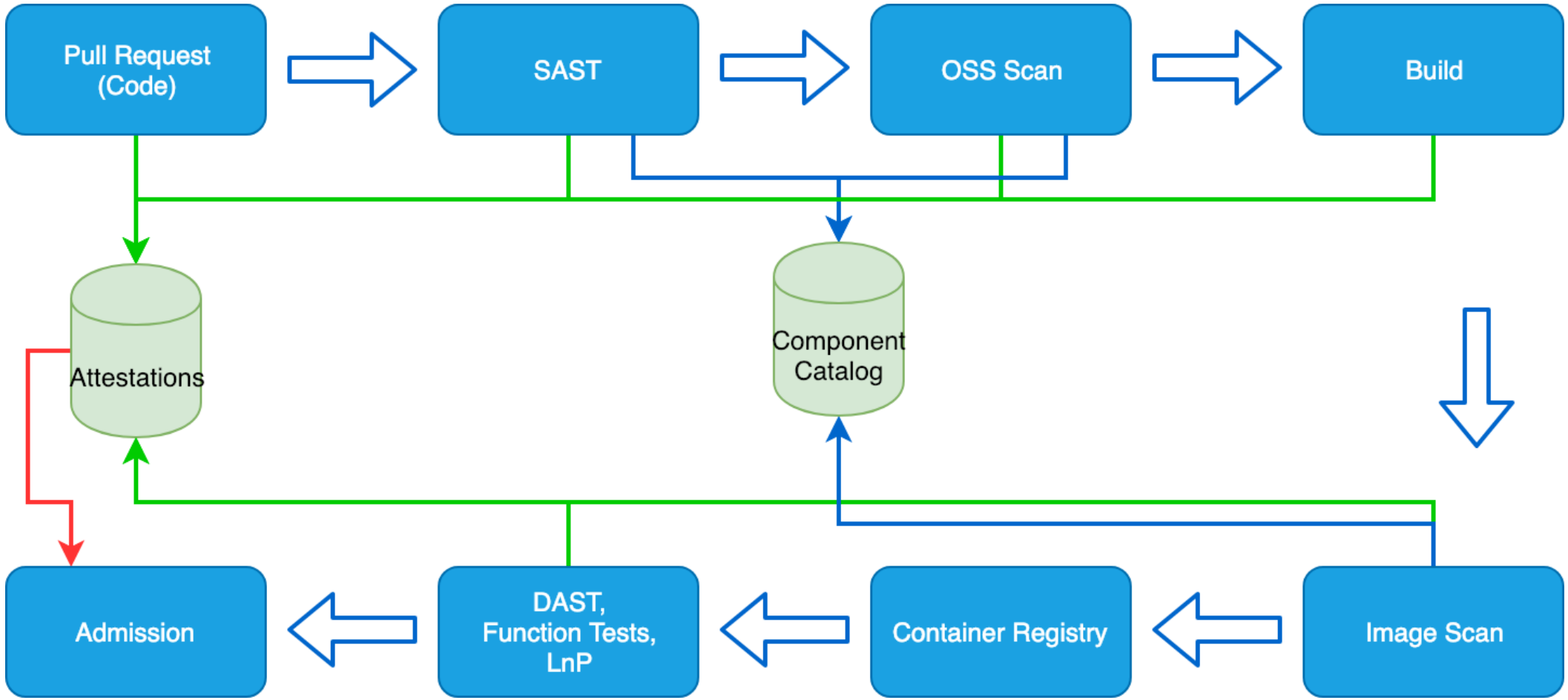
- Harden Kubernetes Control Plane



- Harden Kubernetes Platform



Secure Infrastructure | Build | Ship | Admission | Run



Isolation

- POD Security Policies
- Sandboxing (Kata, gVisor)
- Node Restrictions

Segmentation

- Network Policies
- Zone based clusters, namespaces, Data-classification

Access Control

- RBAC
  - Namespace scoped
  - Cluster scoped

Policies

- Admission Controllers
- Image and resource policies
- Stack specific policies and exceptions

- POD Security Policy

```
spec:
  privileged: false
  allowPrivilegeEscalation: false
  requiredDropCapabilities:
    - ALL
  volumes:
    - 'configMap'
    - 'emptyDir'
    - 'projected'
    - 'secret'
    - 'downwardAPI'
  hostNetwork: false
  hostIPC: false
  hostPID: false
  runAsUser:
    rule: 'MustRunAsNonRoot'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'MustRunAs'
    ranges:
      - min: 1
        max: 65535
```

POD Security Policy (PSP) can be authorized via RBAC. POD will not be created if PSP authorizations are missing for service account used for creating POD.

- Reducing attack surface with Network policy
  - Scenario: Restrict access to java web-app from nginx

Create namespace for Java web-app

```
kubectl create namespace listitem
kubectl label namespace/listitem purpose=catalog
```

Create namespace for nginx

```
kubectl create namespace rproxy
kubectl label namespace/rproxy purpose=frontend
```

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-allow-fe
spec:
  podSelector:
    matchLabels:
      app: web-app
  ingress:
    - from:
      - namespaceSelector:
        matchLabels:
          purpose: frontend
```

- Use WAF for securing applications
  - OWASP
  - Encrypt the traffic to/from POD
  - Enforce Authentication and Authorization

Istio (Service Mesh) is a good candidate

- Detection and Control
  - Privilege escalation detection
  - Container monitoring (cAdvisor)
  - Network detection and controls (Edge security, IDS, Sflow, EBPF)
  - Network Inspection and Visualization

Application threat model

- You need threat-modelling for your applications
- S.T.R.I.D.E is a very useful methodology in modeling threats for applications
  - Analyze and prioritize security initiatives

Threat	Description	Breaks
Spoofing	Pretending to be someone else	Authentication
Tampering	Modifying data that should not be modifiable	Integrity
Repudiation	Claiming someone didn't do something	Non-repudiation
Information Disclosure	Exposing information	Confidentiality
Denial Of Service	Preventing system from providing service	Availability
Elevation Of Privileges	Doing things that one is not supposed to do	Authorization

Final thoughts

- Simplicity is key to success at scale
- Change is inevitable
  - Track technology landscape and associated vulnerabilities
- Empower application developers with knowledge, tools and responsibility
- Prepare Incident response plan
  - Mitigation of control gaps is never sufficient, infrastructure will always have gaps and zero day vulnerabilities

RSAConference2019

Q & A

RSAConference2019

Thank you!