Secure Self-Service Kubernetes

nirmata

Problem: Management Complexity

Day 0

Planning & Design

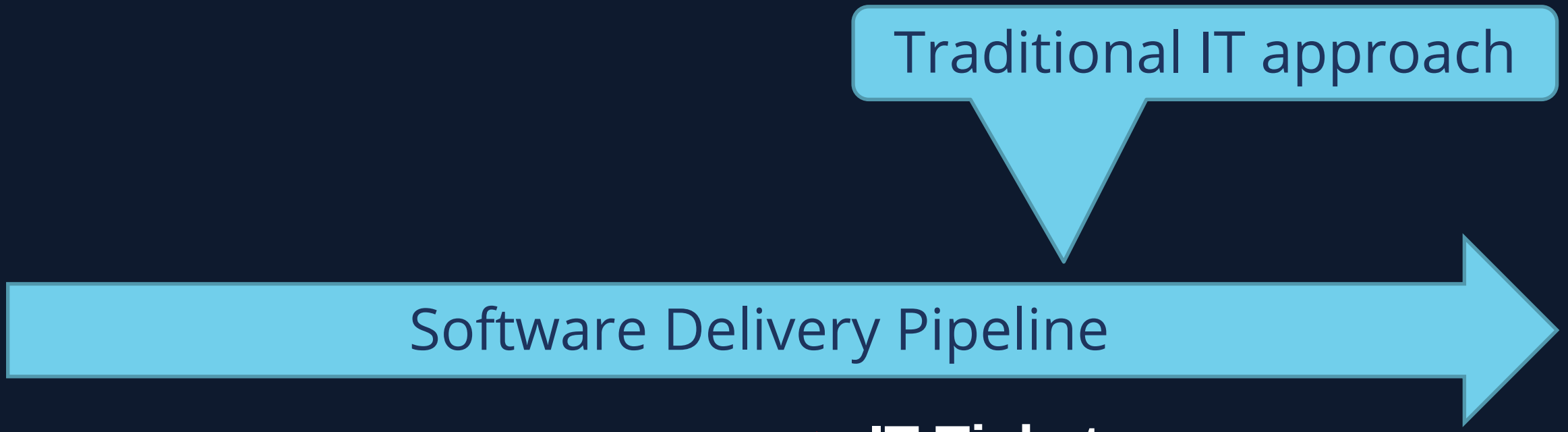
Day 1

Configuration & Deployment

Day 2

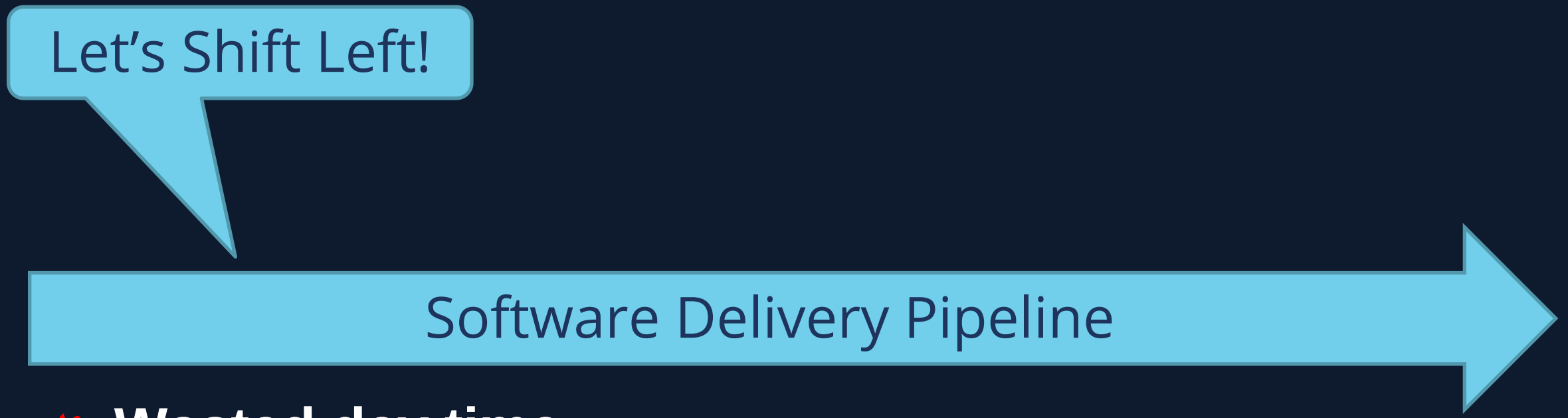
- ✗ Insecure configurations
- ✗ No visibility & governance
- ✗ Kubernetes sprawl

This no longer works…



- ✘ **IT Tickets**
- ✘ **Long wait times**
- ✘ **Resource hugging**

This has its own challenges!



- ✘ **Wasted dev time**
- ✘ **Ad-hoc Security**
- ✘ **Resource sprawl**

The Goal DevOps Harmony

Developers

- Speed
- Agility
- Flexibility

Operators

- Governance
- Compliance
- Costs

What's needed?

1. Workload policy management
2. Virtual Clusters
3. Add-on service management

Jim Bugwadia

- Founder at Nirmata
- Contributor in Policy and Multi-tenancy working groups
- Developer: Golang, Java, JS

@JimBugwadia

Cluster Sizing

What is the best way to manage clusters?

One Cluster per App?

One Cluster per Team?

Shared clusters?

Single Use Clusters

Advantages:

- Teams can manage their own clusters
- Perceived to be easier

Disadvantages:

- Inefficient resource usage
- More clusters to secure and manage

Shared Clusters

Advantages:

- Improved resource utilization
- Separation of concerns (Dev and Ops)

Disadvantages:

- Requires central Ops / Platform team
- Complex to configure and maintain

Considerations

1. Sharing clusters is smart but requires automation.
2. Multiple clusters are inevitable.
3. All clusters needs to be secured and properly configured

How can we enable self-service for dev teams and also ensure security and best practices compliance?

Workload Policies

Workload Policies

- Secure configurations
- Promote best practices

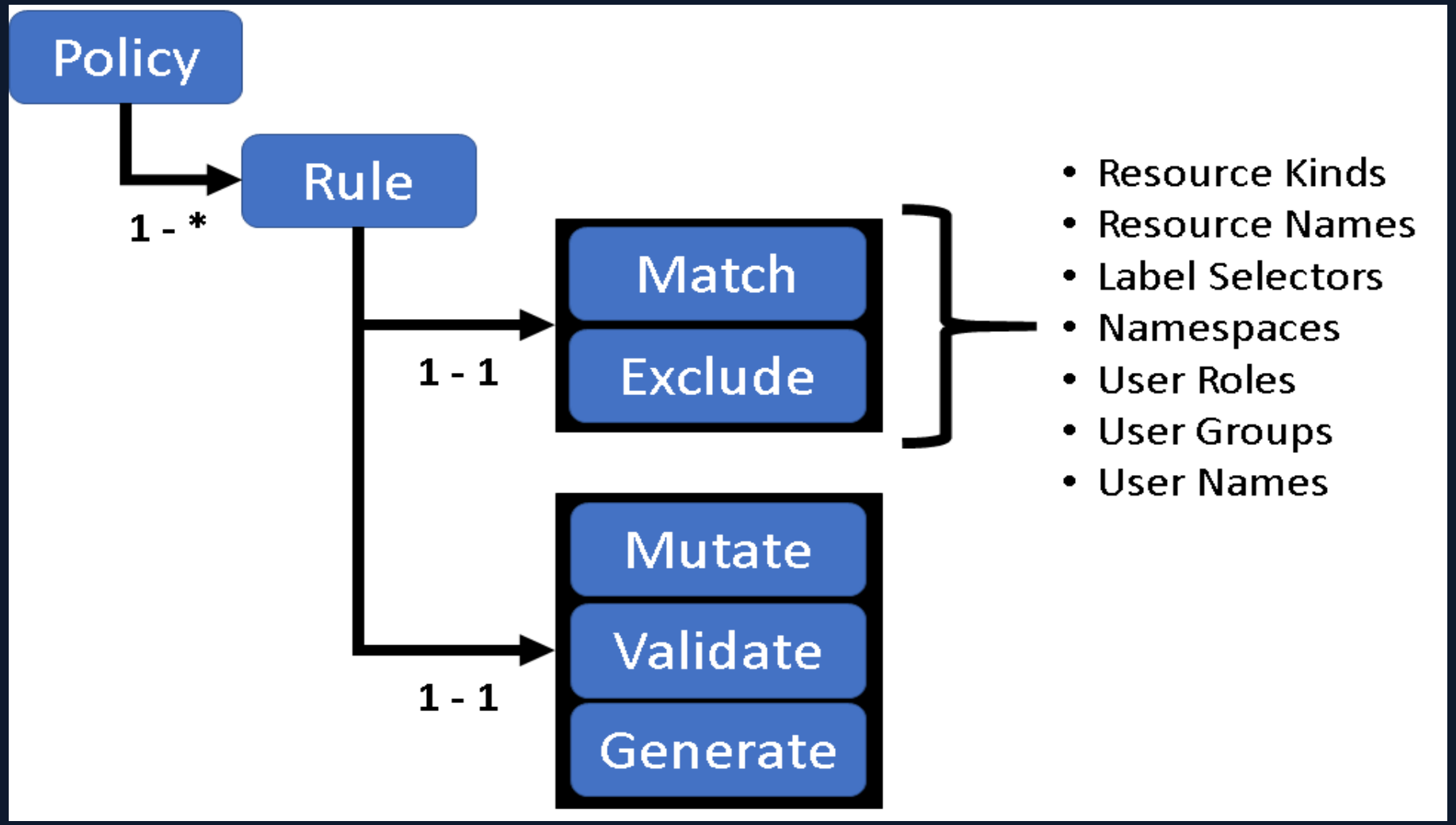
Integrations with all phases of software delivery

- CI/CD pipeline
  - Generate reports and status during builds
- Admission Controls
  - Block insecure or non-compliant configurations
- Background scanning
  - Periodically scan and report violations and configuration errors

Kyverno OSS Kubernetes Policy Engine

- Kubernetes Policies as declarative configuration
  - Validate using overlays
  - Mutate using JSON patch and Strategic Merge Patch
  - Generate and synchronize across namespaces
- Admission controls for in-cluster checks
- Periodic jobs and reporting
- CLI for offline checks

Kyverno Policy Model



Sample Policy: Validate Pod User

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: disallow-root-user
spec:
  rules:
  - name: validate-runAsNonRoot
    match:
      resources:
        kinds:
        - Pod
    validate:
      message: "Running as root user is not allowed. Set runAsNonRoot to true"
      anyPattern:
      - spec:
          securityContext:
            runAsNonRoot: true
      - spec:
          containers:
          - securityContext:
              runAsNonRoot: true
```

OPA or Kyverno?

```
package k8spspreadonlyrootfilesystem

violation[{"msg": msg, "details": {}}] {
    c := input_containers[_]
    input_read_only_root_fs(c)
    msg := sprintf("only read-only root filesystem container is allowed: %v", [c.name])
}

input_read_only_root_fs(c) {
    not has_field(c, "securityContext")
}
input_read_only_root_fs(c) {
    has_field(c, "securityContext")
    not has_field(c.securityContext, "readOnlyRootFilesystem")
}

input_containers[c] {
    c := input.review.object.spec.containers[_]
}
input_containers[c] {
    c := input.review.object.spec.initContainers[_]
}

# has_field returns whether an object has a field
has_field(object, field) = true {
    object[field]
}
```

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: require-ro-rootfs
spec:
  rules:
  - name: validate-readOnlyRootFilesystem
    match:
      resources:
        kinds:
        - Pod
    validate:
      message: "Root filesystem must be read-only"
      pattern:
        spec:
          containers:
          - securityContext:
              readOnlyRootFilesystem: true
```

Kyverno

Watch the video review on Coffee and Containers:

"Ultimately I decided that this is a super cool tool!“

-- Adrian Goins, Director of Community and Evangelism, Rancher Labs

https://www.youtube.com/watch?v=DW2u6LhNMh0&feature=youtu.be&t=76

- Try Kyverno (and ★) at: https://kyverno.io
- Say “hello!” and get involved: Kubernetes Slack #kyverno

Virtual Clusters

The Problem

- Kubernetes was designed to efficiently bin-pack workloads across pools of infrastructure
- But Kubernetes is complex to configure and secure
- This complexity has led to cluster sprawl
  - Easier to get going but can lead to even greater complexity!

Virtualization

- A software-defined form of a physical resource
- Typically make things easier to use and secure
- Enables new usage models (developer sandboxes!)

Easy	Hard
Virtual Machine	Physical Machine
Virtual Network	Physical Network
Virtual Storage	Physical Storage

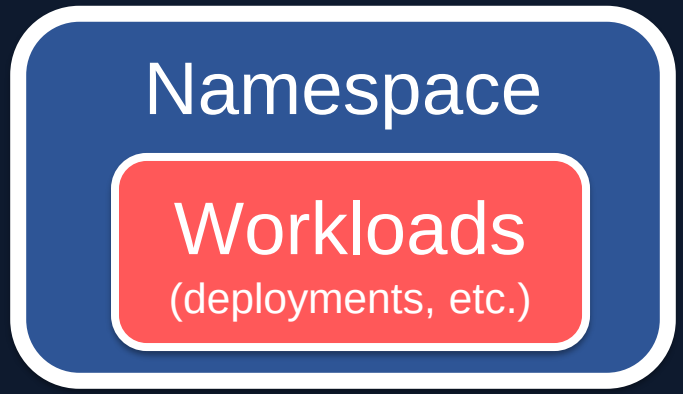
Kubernetes Virtualization

1. Namespaces based
   - suitable for teams within an enterprise
2. Control & data-plane based
   - suitable for service providers with multiple tenants

*This presentation will focus on the 1st – for more details on both models visit and join the **Kubernetes Multi-Tenancy Working Group!***

https://github.com/kubernetes-sigs/multi-tenancy

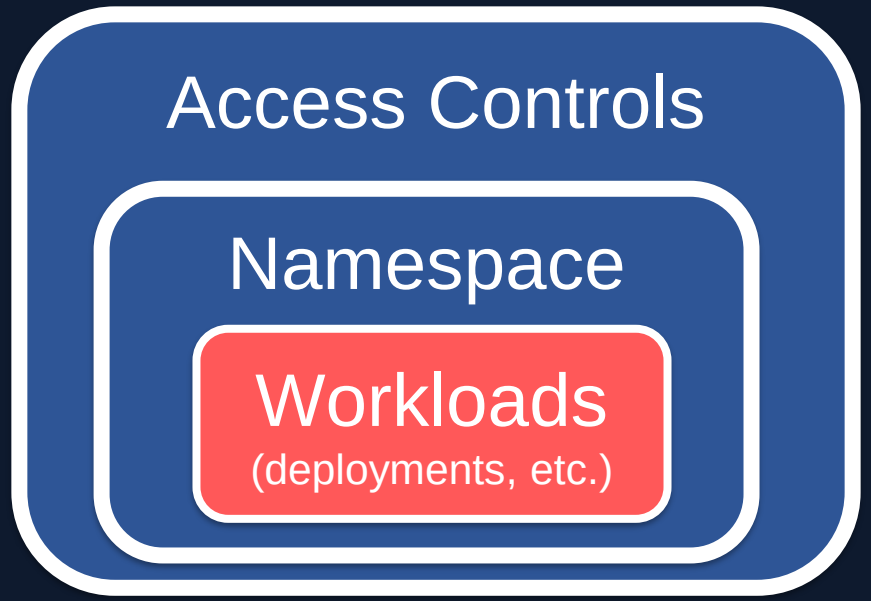
Namespaces Based Virtualization



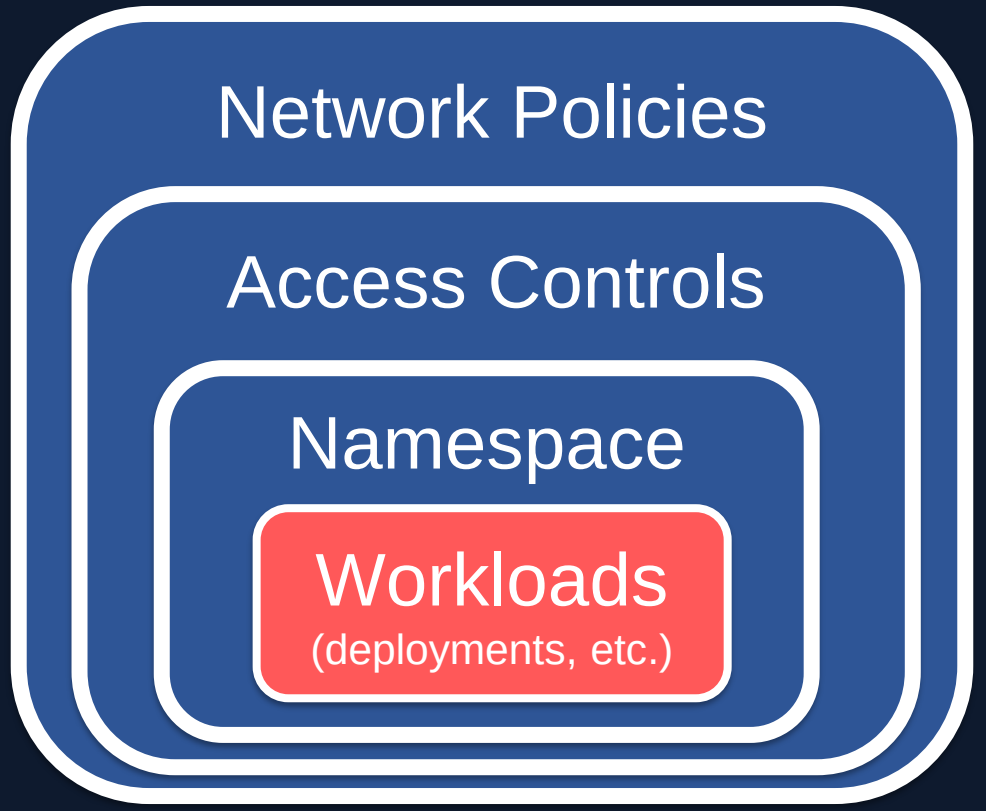
Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called namespaces.

https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

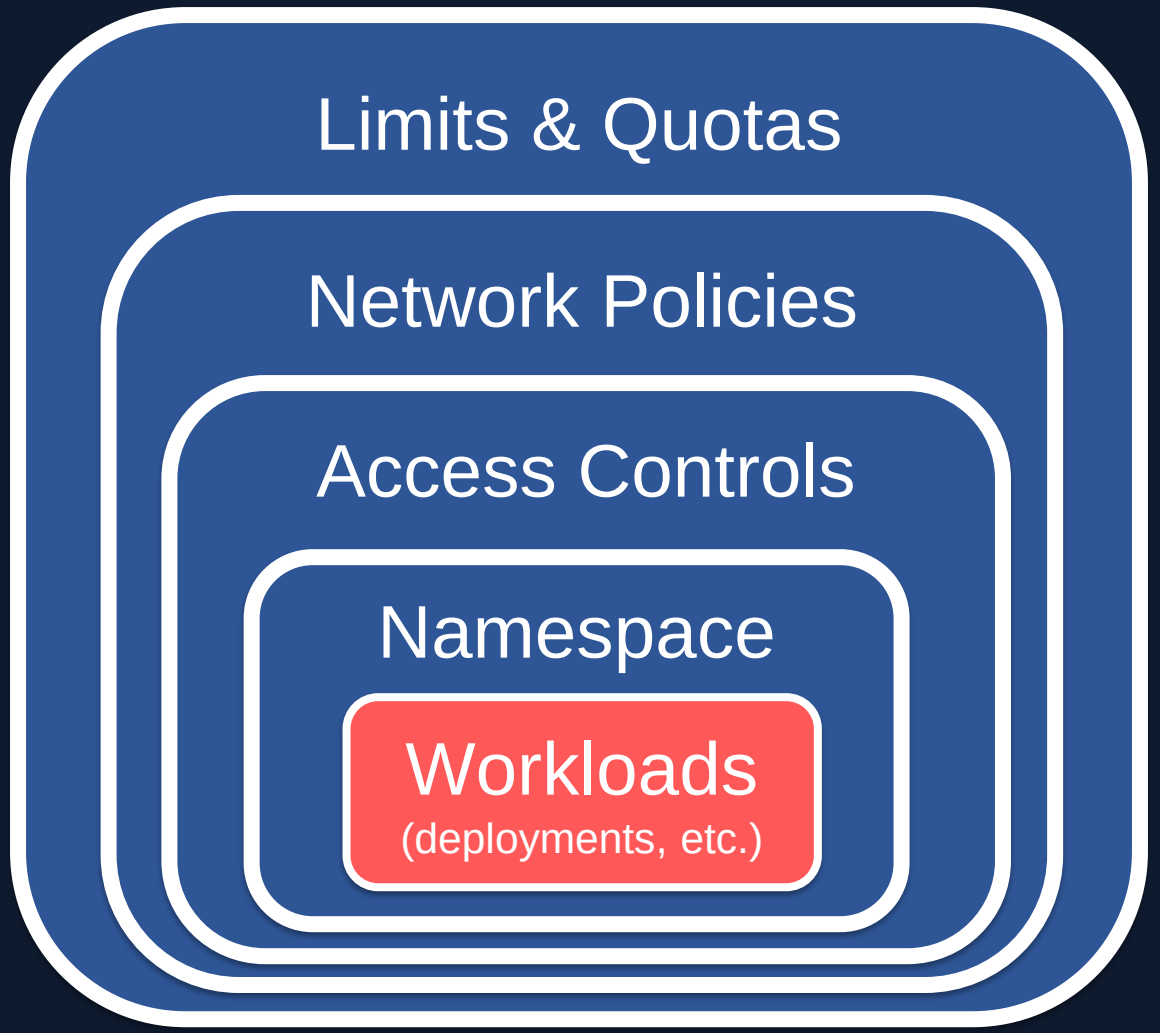
Namespaces Based Virtualization



Namespaces Based Virtualization

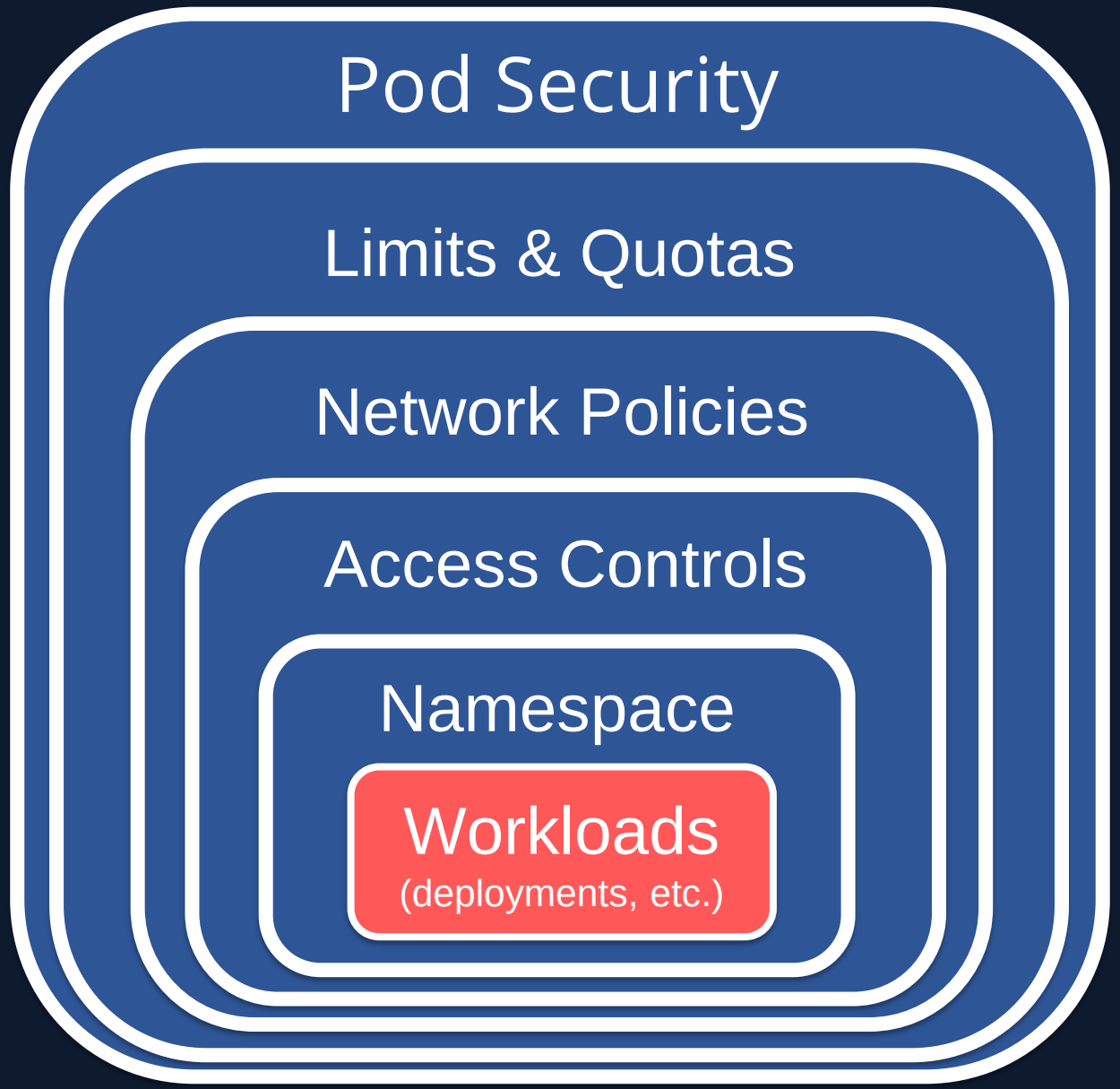


Namespaces Based Virtualization



nirmata

Namespaces Based Virtualization



nirmata

Namespaces Based Virtualization

- Yes, this may seems complex...
- But , luckily all of this can be automated!
- And, there is a way to measure multi-tenancy!

Multi-Tenancy Benchmarks

Guidelines for securely sharing clusters within an enterprise and tools to measure conformance

https://github.com/kubernetes-sigs/multi-tenancy/tree/master/benchmarks

Add-on Services

Add-On Services

Cluster add-ons

- Shared services used across tenants
- Must be installed before any workloads are deployed
- Lifecycle and upgrades managed by central team
- Security, monitoring, logging, service mesh, etc.

Add-On Services

Environment add-ons

- Services used by tenants for their own apps or workloads
- Self-service lifecyle managed by app owner
- Common catalog or repository to help standardize
- Application monitoring, logging, ingress, etc.

Challenges with Add-On Services

1. Who manages these?
2. How do you know what is deployed and where?
3. How do you update across all clusters?

Summary

Key Takeaways

- Enabling Secure Self-Service Kubernetes requires:
  1. Workload Policies
  2. Virtual Clusters
  3. Add-on Services Management

Focus on delivering value

- Kubernetes management tools, like Nirmata, exist to solve these issues…use them!

"You cannot reduce complexity [in systems]. The best you can do is manage it." - Grady Booch

Next steps...

- Join the Multi-Tenancy WG: kubernetes-sigs/multi-tenancy
- Try Kyverno (and ★) at: https://kyverno.io
- Try Nirmata for free at: https://try.nirmata.io
- Contact me at: @JimBugwadia

Thank-You!

https://www.nirmata.com