Access Control In Kubernetes

What's Missing, And How To Fix That

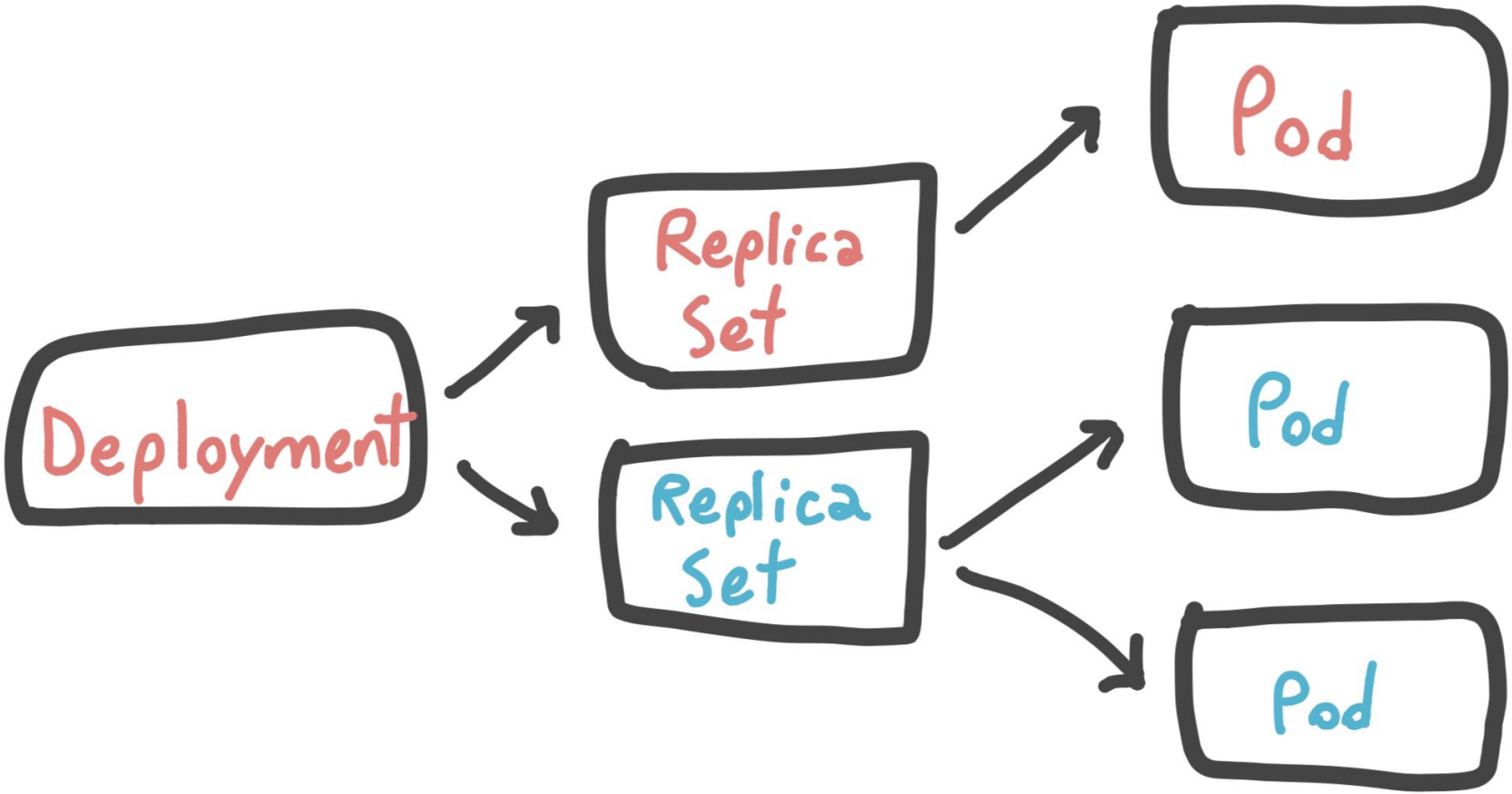
@vllry

@SethMcCombs

Disclaimer

Opinions!

- This is not about Lyft systems
- This is not about Triller systems
- We might be wrong


Deployment
Replica Set
Replica Set
Pod
Pod
Pod

What Can We Do About All This?

- RBAC
- Network Policies (native, Calico, Istio, etc)
- Mutating admission webhooks (native)
- Open Policy Agent
- Custom API gateways

Broad Categories

- Kubernetes access
  - What can the app do to the Kubernetes control & data planes?
- Runtime characteristics
  - What can run, and in what way?
- Network access
  - What can access what?

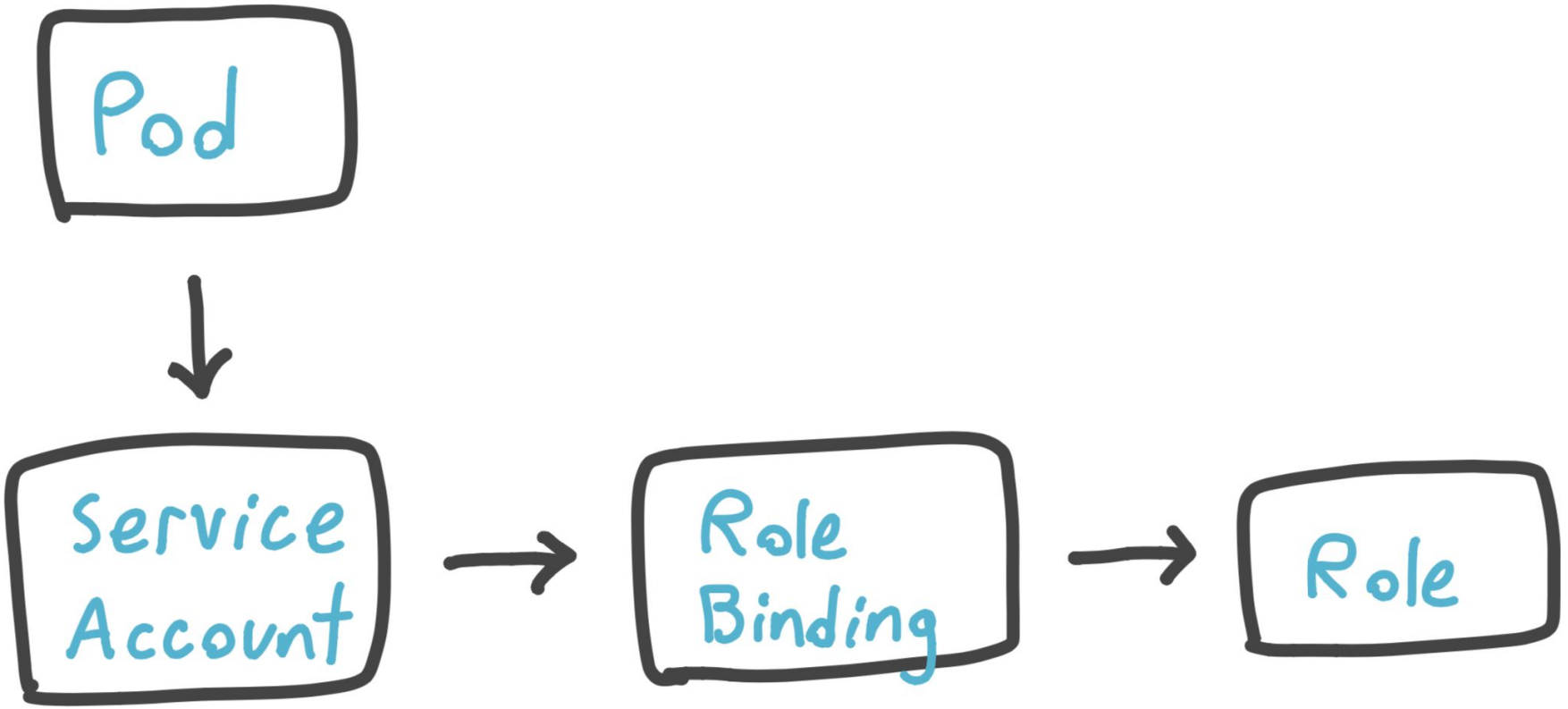
RBAC

RBAC Role

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

RBAC RoleBinding

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: vallery
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```


Pod
Service Account
Role Binding
Role

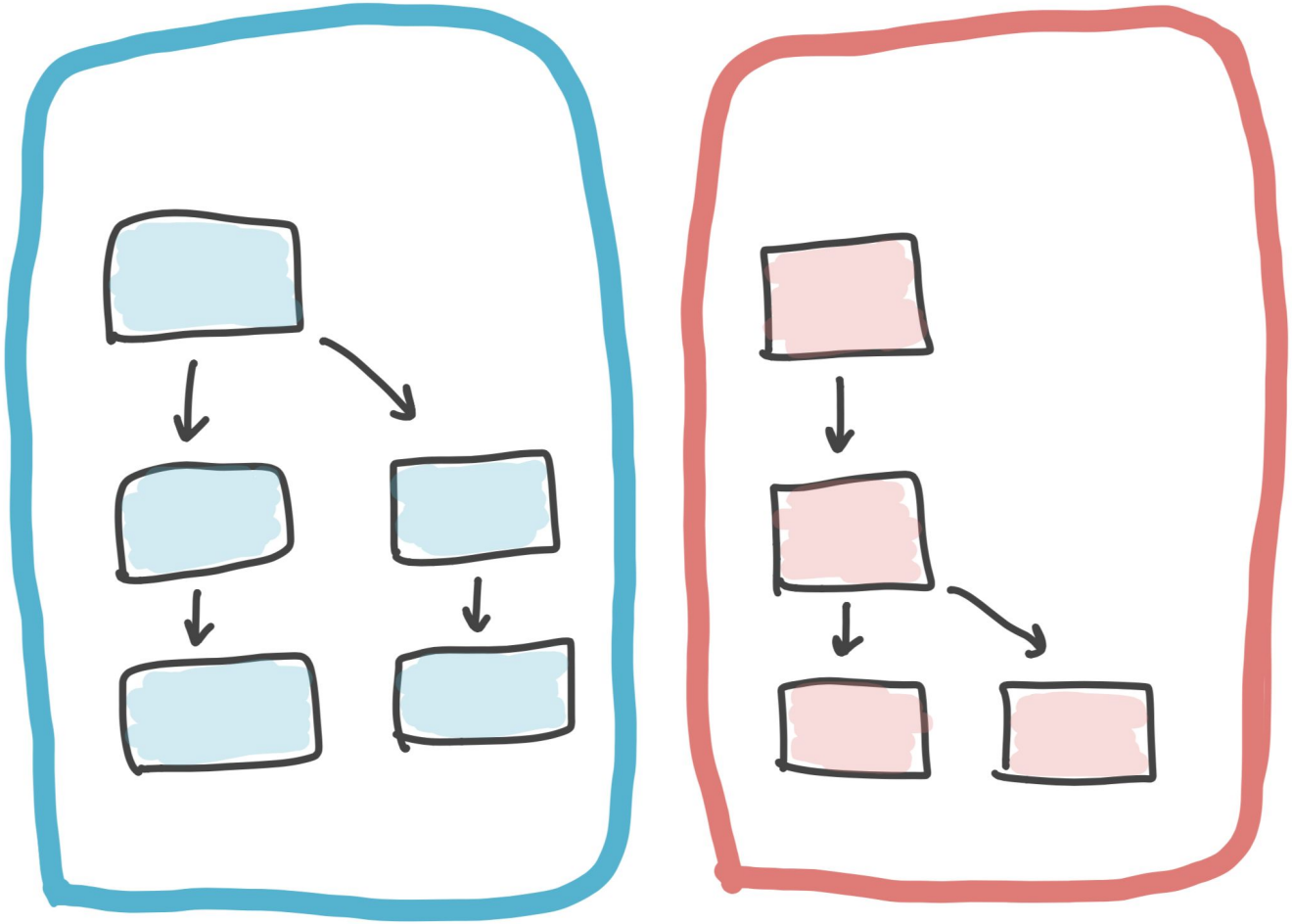
Other Forms of K8S API Access Control

- ABAC: granular user-based permissions.
- Node: for kubelets only. Kubelet is granted permissions based on the pods it runs.
- Webhook: post to URL, use that response.

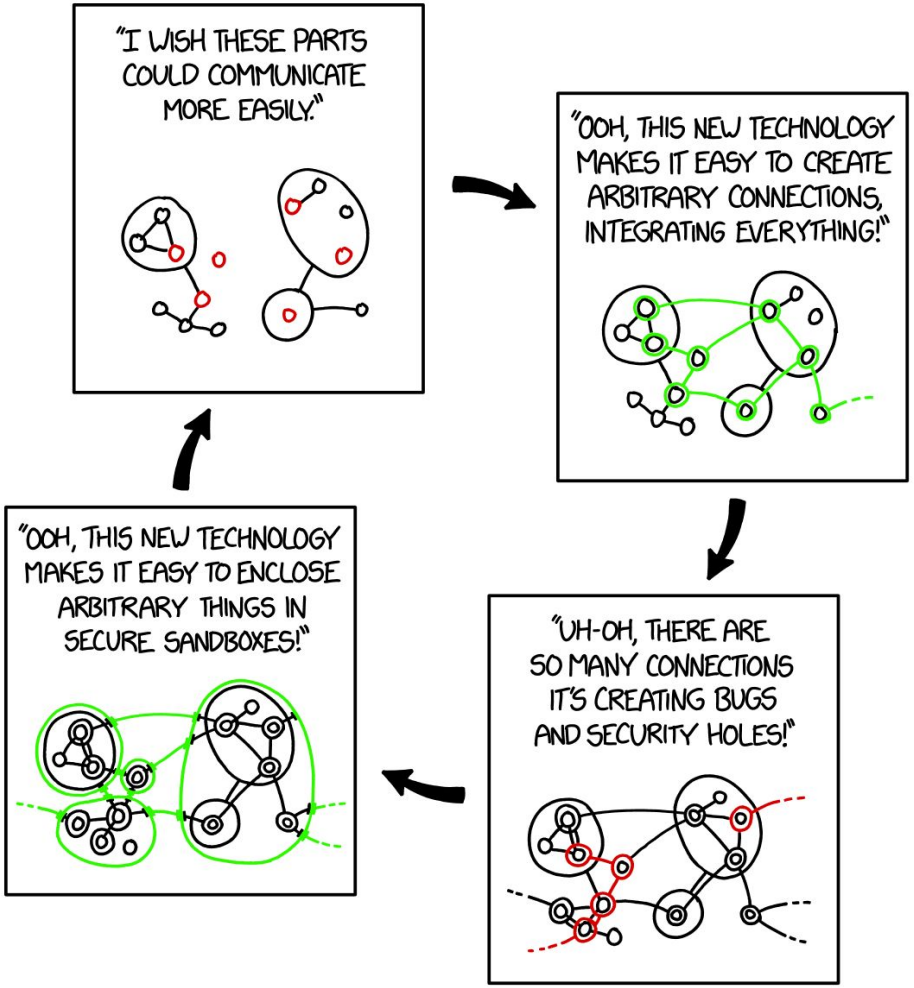
RBAC Limitations

- No sub-object access control
- Universal permission by resource type

Namespaces



“I wish these parts could communicate more easily.”



https://xkcd.com/2044/

Admission Webhooks

Admission Webhooks

- Validating/Mutating AdmissionWebhooks used to verify/modify Kubernetes objects declared via the Kubernetes API
- Custom hooks can be written to ensure resources not meeting cluster criteria are not created - resource creation denied
- Change objects missing certain requirements - Adding labels/annotations, resource limits, etc
- Hooks can be used to restrict the creation of objects in a Namespace, but aren’t actively controlling what runs in the namespace after `create`/`apply`

Custom API Gateways

What is a gateway/deputy?

- A deputy/gateway is designed to perform specific actions, which require elevated permissions.
- The deputy exposes an API to trigger these actions.
- Acts as a *logical gate* to the underlying system.

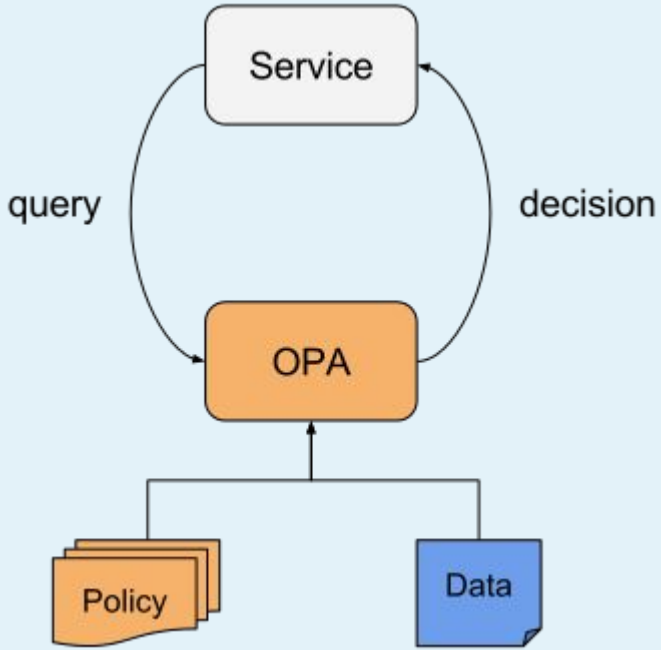
API Deputy Drawbacks

- You still have a service with elevated permissions.
- Many actions require a very *simple* logical gate.
  - EG “only allow updates to this field”.

Open Policy Agent (OPA)

Open Policy Agent

- OPA offloads Policy Decisions from a service (across the stack)
- Queries - whether an action can be taken by a service, answer is provided back to the service with an allow or deny.
- Allows context specific based on status or data of other services in the system



© 2019 Open Policy Agent contributors

Using OPA for Resource Access Control

- Calls to Kubernetes API are sent to OPA with the JSON Object
- OPA compares to its rules, and returns an Allow or Deny (Validating AdmissionWebhook)
- OPA can tell you the reason *why* the action was not allowed
- Can also return a JSON patch to modify the object, thus acting as a Mutating AdmissionWebhook

Using OPA for NetworkPolicy

- OPA could be deputized to provide context specific updates/changes to Kubernetes NetworkPolicy, taking advantage of a CNI Plugin like Calico, Cilium or others
- Labels or Annotations applied sets of objects could be used to manage intra-Namespace communication between pods, services, etc.
- The combination of OPA for NetworkPolicy and AdmissionControllers would allow only properly annotated resources to be created, and those annotations would further trigger creations/updates on NetworkPolicy rules

Network Access Policy

Service-Level Network Perimeters

- The most convenient way to restrict access is to outright block network traffic.
- Not part of the Kubernetes network model.

Calico (Review for inclusion)

- Restrict network ingress, by origin.
- Restrict network egress, by origin.

Calico (Examples if needed)

Deny All Egress

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-egress
  namespace: advanced-policy-demo
spec:
  podSelector:
    matchLabels: {}
  policyTypes:
   - Egress
```

Deny all Ingress

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-ingress
  namespace: advanced-policy-demo
spec:
  podSelector:
    matchLabels: {}
  policyTypes:
   - Ingress
```

Allow Ingress to pod matching Label (from pod matching a label?)

```
apiVersion:
networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-nginx
  namespace:
advanced-policy-demo
spec:
  podSelector:
    matchLabels:
      $KEY: $VALUE
  ingress:
    - from:
      - podSelector:
          matchLabels:
           $KEY: $VALUE
```

DNS Traffic Egress

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-dns-access
  namespace: advanced-policy-demo
spec:
  podSelector:
    matchLabels: {}
  policyTypes:
  - Egress
  egress:
  - to:
    - namespaceSelector:
        matchLabels:
          name: kube-system
    ports:
    - protocol: UDP
      port: 53
```

Time For The Opinions

Optimizing Boundaries

Sub-namespace Permissions?

```
SpaceshipGrey:~ vallery$ ls -l
total 32
drwx------@   5 vallery  staff    ... Applications
drwx------+  17 vallery  staff    ... Desktop
```

```
apiVersion: ...
kind: ...
metadata:
    ...
spec:
    ...
status:
    ...
```

```
apiVersion: ...
kind: ...
metadata:
    owningRoles:
        - jenkins
        - ops
    ...
spec:
    ...
status:
    ...
```

Smaller Namespaces?

What Are Our Limiting Factors?

- Objects that rely on one another need to be in the same namespace.
  - EG Ingress / Service / Deployment, HPA / Deployment
- (Namespace level) accounts can’t be used in multiple namespaces.
  - Users / bots will need more accounts.

Too Long, Didn’t Listen

1. Use RBAC roles with namespaces to segment access.

2. Reduce network access between pods, with a service mesh or policy tool.

3. Gate complex, high-access behavior behind APIs and controllers (off the shelf, or bespoke).

Vallery Lancey

- Infrastructure at Lyft
- @vllry on most things

- Kubernetes contributor (mostly SIG-network, but it's scattered). Dabbles in many aspects of distributed systems.

Seth McCombs

- SRE at Triller
- Tweets at @SethMcCombs

- More ops than dev, fiddles with containers and Kubernetes. Avid guitarist and collector of fountain pens.