Kubernetes Workshop

Kubernetes Training

8: Workload CI/CD and Management

INNOQ

Organization and Security

Organization and Security

Organize Apps / Workloads / Projects / Teams with:

- Namespaces
- Labels
- Role Based Access Control
    - User Roles and Rights
    - Service Accounts
- Network Policies
    - Networking between PODs
- Pod Security Policies

Network Policies

- Controls POD networking
- Controls Namespace networking

Namespaces and Labels

- Namespaces
  - are intended for use with multiple teams, or projects
  - provide a scope for names
  - way to divide cluster resources between multiple uses
- Labels
  - separate slightly different resources
  - different versions of the same software
  - Separate different SCS, Microservices etc

Label Examples

```
labels:
    component: elasticsearch-log
    controller-revision-hash: es-data-log-distributed-4183403337
    name: es-data-log-distributed
    role: data
```

```
$ kubectl get po -l role=data --all-namespaces
NAMESPACE    NAME                        READY     STATUS     RESTARTS   AGE
monitoring   es-data-log-distributed-0   1/1       Running    0          22h
abc-defpro   es-data-AAA-distributed-0   1/1       Running    0          22h

$ kubectl -n monitoring get po -l component=elasticsearch-log
NAME                             READY     STATUS     RESTARTS   AGE
es-client-log-3534390662-1vhz0   1/1       Running    1          22h
es-data-log-distributed-0        1/1       Running    1          22h
es-master-log-3131274025-5nfz3   1/1       Running    1          22h
```

Exercise Labels

1. *Do you find all scs named foobar?*
2. *Play around with labels and sets and --all-namespaces*

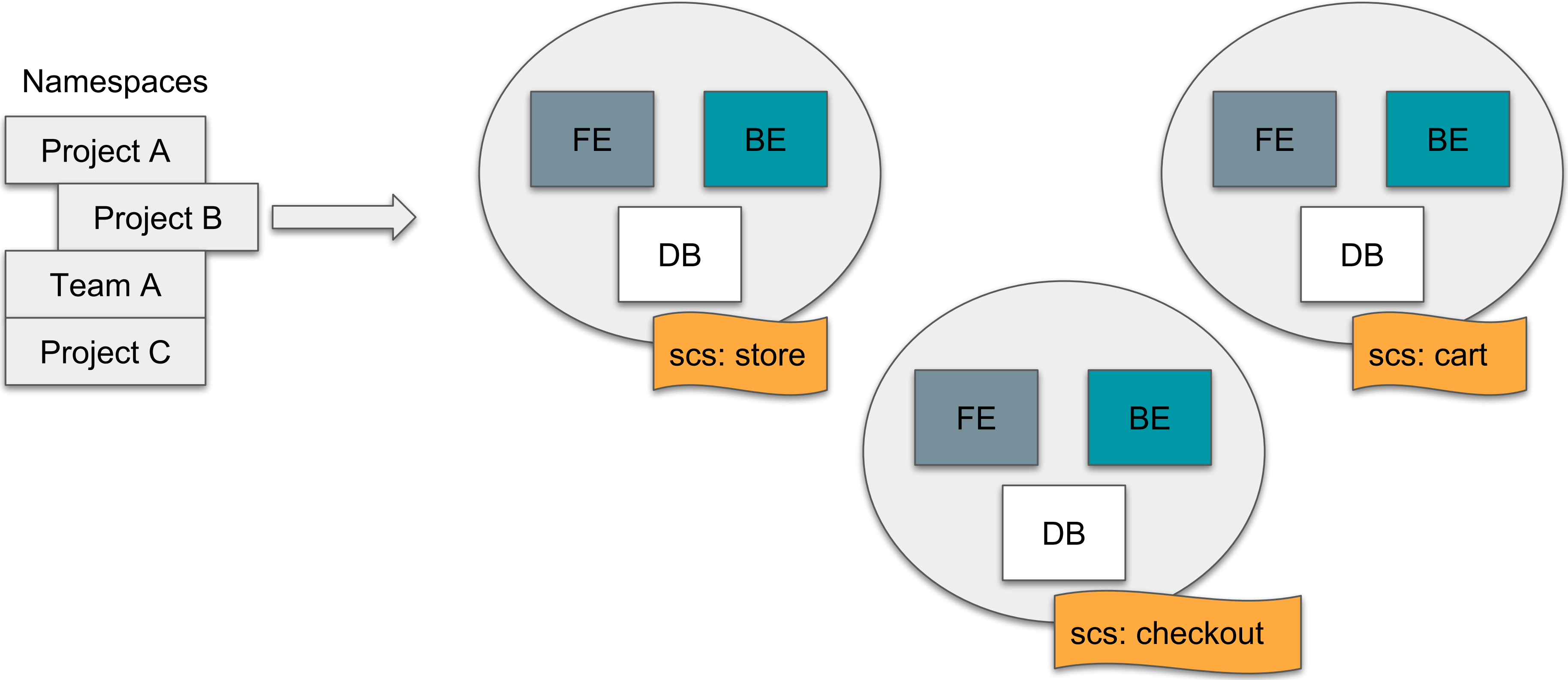
```
kubectl get pods -l 'role in (data),scs in (foobar)'
```

Label Examples

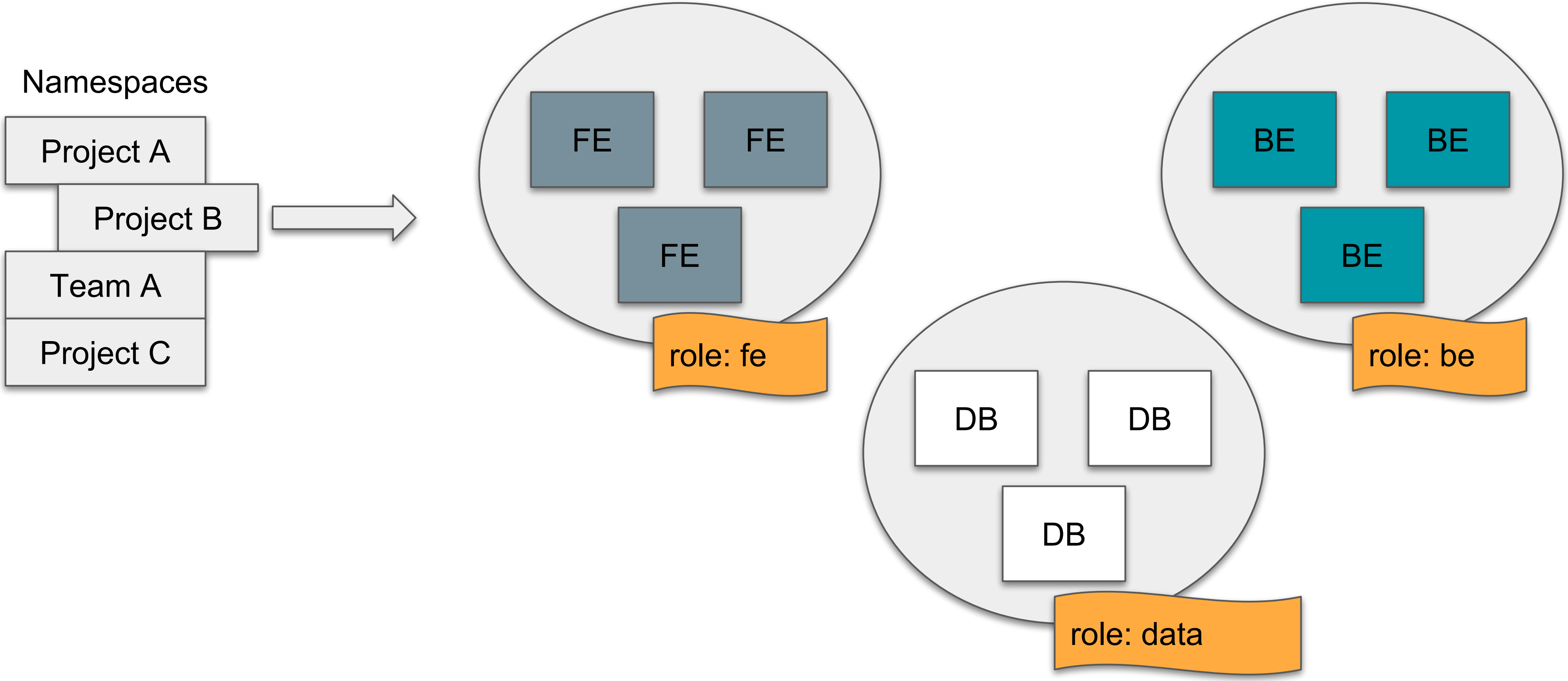
Doing so, you can:

- Select a specific component (service) in a namespace (project)
- Select a specific role in a namespace
- Select all specific roles
- Select all specific components
- . . .

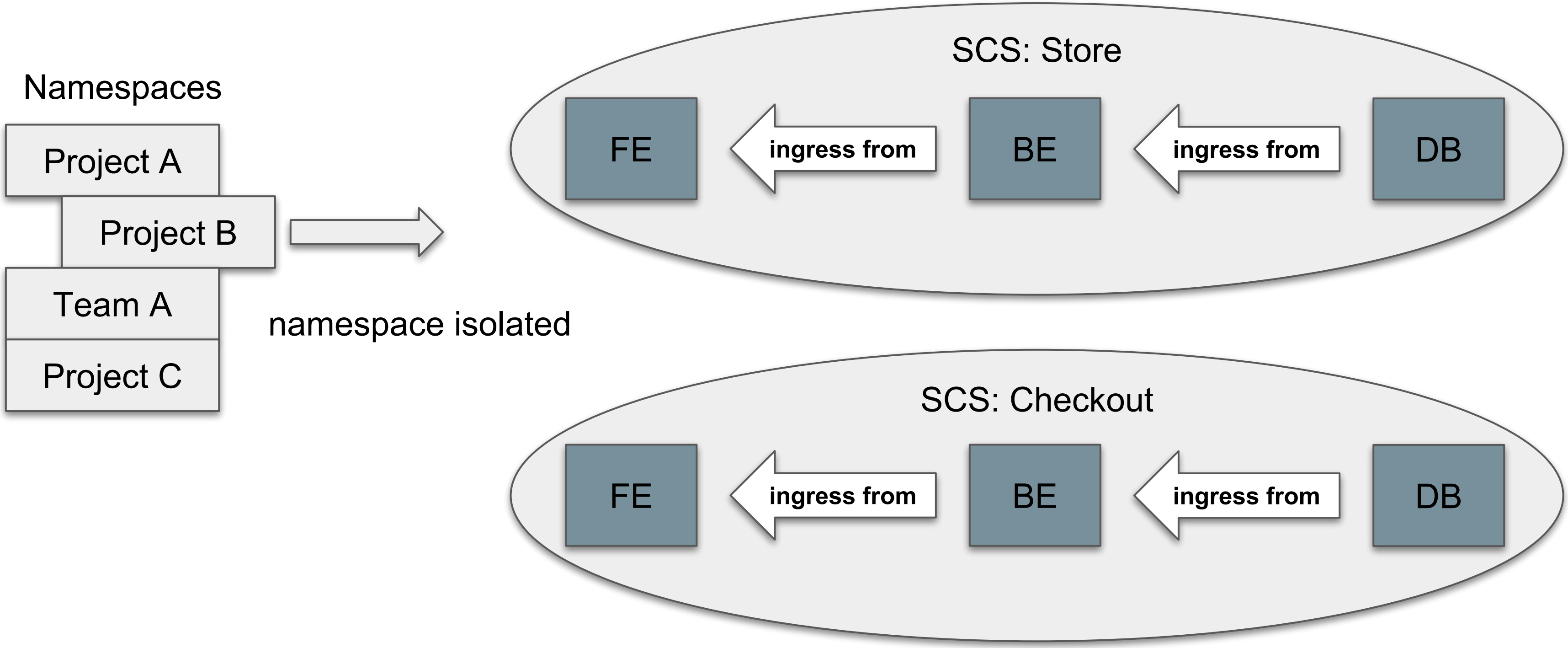
Service (SCS) Labels



Role Labels


Namespaces
Project A
Project B
Team A
Project C
FE
FE
FE
role: fe
BE
BE
BE
role: be
DB
DB
DB
role: data

Network Policies



Managing CI/CD

Complex Workload Types

- Static Deployments (simple scaling)
  - `kubectl apply -f …`
  - `kubectl scale --replica=3 statefulset …`
  - ➪ use native YAMLs or Helm Charts

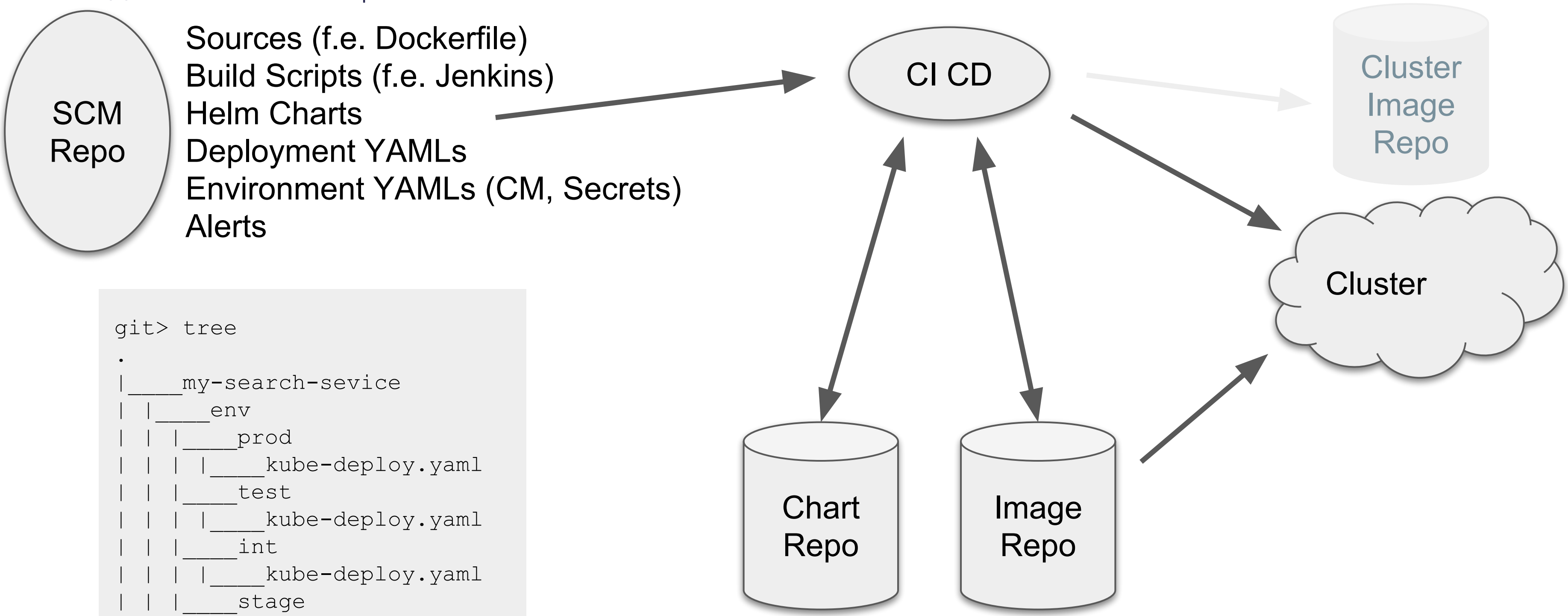
- Adoptable Deployments (scaling or changes are complex)
  - `kubectl apply -f my-postgresql-crd.yaml`
  - ➪ use/write an Operator/Controller

Deployment Ingredients

- Source Repository
- Deployment Scripts (Env independent)
- Image Repo
- Helm Script Packager (incl. Chart Repo)
- Configuration (Env dependent)
- Secrets (TLS, Passwords etc. env dependent)
- RBAC for Service Accounts
- Things like Prometheus Alerts

General Repository Rules

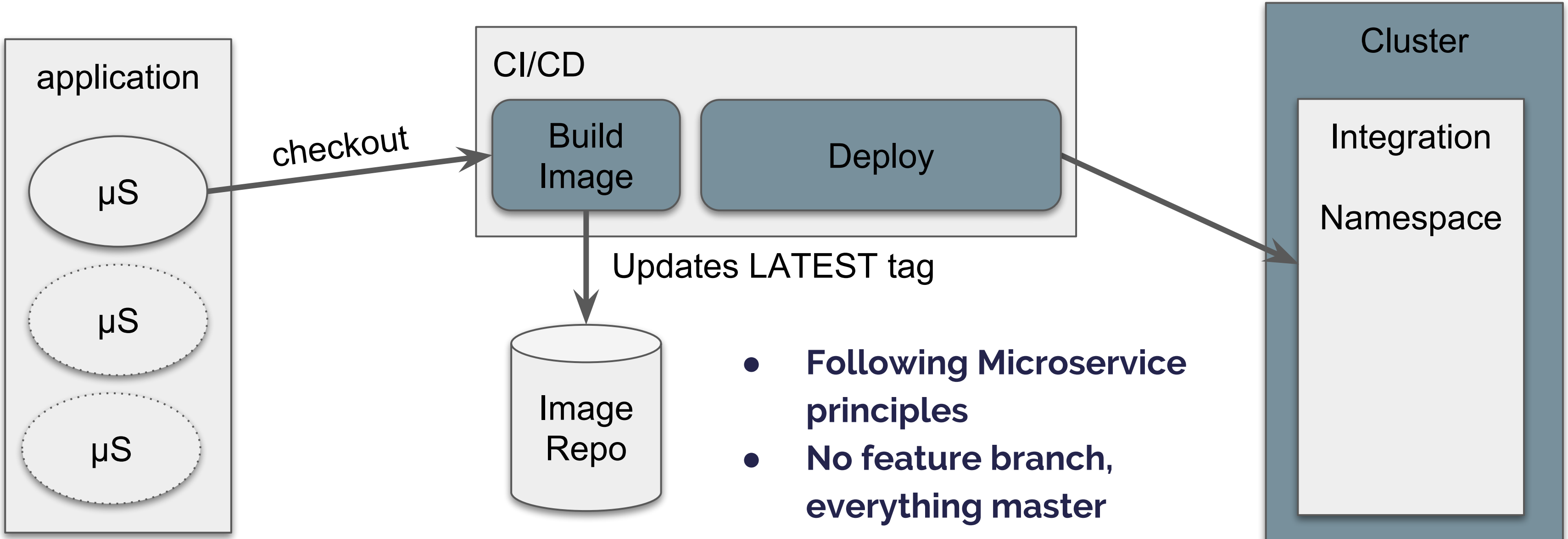
- One SCS may consist of several Microservices
- Microservices are deployable independently
- ➪ One SCM Repository per Microservice
- ➪ All needed to …
  - Build
  - Deploy
  - Monitor / Alert
  - Configure
- … belongs to this Repo



Sources (f.e. Dockerfile)
Build Scripts (f.e. Jenkins)
Helm Charts
Deployment YAMLs
Environment YAMLs (CM, Secrets)
Alerts

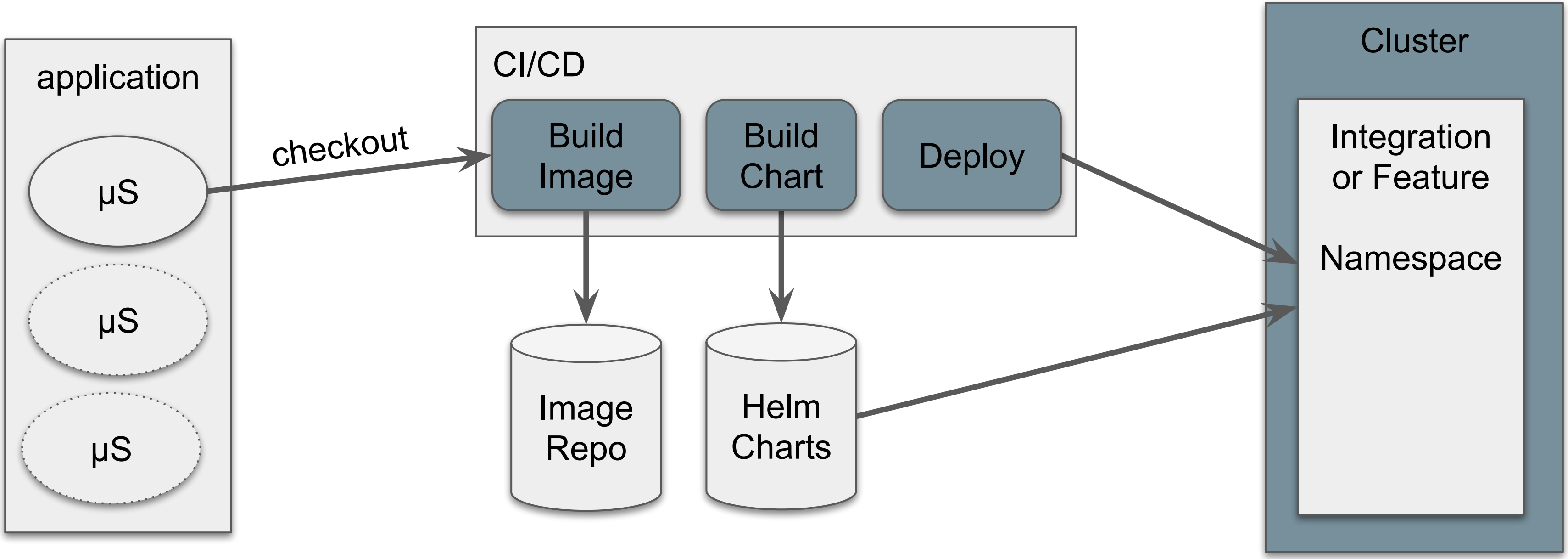
```
git> tree
.
|____my-search-sevice
| |____env
| | |____prod
| | | |____kube-deploy.yaml
| | |____test
| | | |____kube-deploy.yaml
| | |____int
| | | |____kube-deploy.yaml
| | |____stage
| | | |____kube-deploy.yaml
| |____subproj1
| | |____Dockerfile
| |____subproj2
| | |____Dockerfile
| |____kube-deploy.yaml
| |____Makefile
```

LATEST tag Deployment



- Following Microservice principles
- No feature branch, everything master
- Deployment-scripts in a dedicated repo (per µS)
- Namespace per Project

Package Build and Deployment


application
µS
µS
µS
checkout
CI/CD
Build Image
Build Chart
Deploy
Image Repo
Helm Charts
Cluster
Integration or Feature
Namespace