Kubernetes 기초

Configuration File(YAML format)
Label, Selector

Kubernetes 의 Configuration File (YAML format)

Cluster에 원하는 서비스를 구동 하는 방법은 이에 대한 원하는 상태를
- **직접 요구하거나**
- **해당 내용을 파일로 정리하여**

APIServer를 통해 **파일에 기술한 정보대로 Master내의 etcd에 원하는 상태정보를 표현한 Object(Resource)를 만들게 하는 것이다.**

YAML(**Y**AML **A**in't **M**arkup **L**anguage)

nginx-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16
        ports:
        - containerPort: 8080
```

- 사람이 읽기 쉬운 데이터 직렬화 양식
- Syntax: strict indentation!(들여쓰기)

YAML VALIDATOR

- https://onlineyamltools.com/validate-yaml

5 Fields of Kubernetes Definition File

- **apiVersion**
  - 오브젝트를 생성하기 위해 사용하고 있는 쿠버네티스 API 버전
- **kind**
  - 어떤 종류의 쿠버네티스 리소스인지 기술
- **metadata**
  - 오브젝트를 구별해줄 수 있는 데이터
  - name, label, namespace, annotations …
- **spec**
  - 오브젝트로 생성하고자 하는 리소스의 구체적인 정보
- **status**
  - 현재 오브젝트의 상태
  - 쿠버네티스에 의해 자동으로 생성

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

<deployment.yaml>

※ 오브젝트: 원하는 상태가 저장된 쿠버네티스 API Server상의 객체

apiVersion

- **apiVersion**
  - 오브젝트를 생성하기 위해 사용하고 있는 쿠버네티스 API 버전
    - v1: 쿠버네티스에서 발행한 첫 stable release API
      - 대부분의 api 포함
      - Pod, Namespace, Node, Service, …
    - apps/v1: 쿠버네티스의 common API 모음
      - Deployment, RollingUpdate, ReplicaSet, …
    - rbac.authorization.k8s.io/v1
      - 쿠버네티스의 role-based access control이 가능한 function 정의
      - ClusterRole, RoleBinding, Role, …
    - …

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

<deployment.yaml>

kind

- **kind**
  - 어떤 종류의 쿠버네티스 오브젝트 리소스인지 기술
    - Pod: 쿠버네티스에서 배포할 수 있는 가장 작은 컴퓨팅 단위
    - ReplicaSet: Pod의 replica 개수 유지를 보장하는 컨트롤러
    - Deployment: Pod와 ReplicaSet에 대한 업데이트를 가능하게 하는 컨트롤러
    - …

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

<deployment.yaml>

metadata

- **metadata**
  - 오브젝트를 구별해줄 수 있는 데이터

    - name: 해당 오브젝트의 이름
    - label: 해당 오브젝트의 label 값
    - namespace: 지정한 네임스페이스에 오브젝트 리소스 생성
    - annotations: 임의의 키/값 을 추가하는 일종의 주석 역할

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

<deployment.yaml>

spec

- **spec**
  - 오브젝트로 생성하고자 하는 리소스의 구체적인 정보
  - 쿠버네티스 리소스마다 요구하는 spec 필드의 정보는 다를 수 있음

  - ex) Deployment
    - selector: matchLabels와 동일한 label값을 가진 리소스를 선택
    - replicas: 리소스를 몇 개의 복제본으로 생성할지
    - template: 해당 오브젝트의 구체적인 형식
      - metadata: 생성할 리소스의 기본 정보
      - spec: 생성할 리소스(ex. Pod)의 구체적인 스펙 정보

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

<deployment.yaml>

Status

- **쿠버네티스에 의해서 자동으로 생성**

- **의도하는 상태(desired status)와 현재 상태(actual status)를 꾸준히 비교하고 업데이트**
  - desired status: 사용자가 배포하고자 하는 쿠버네티스 오브젝트에 대해 원하는 상태
    - yaml 파일을 이용하거나 run/create 과 같은 명령어를 이용하여 명시
  - actual status: 해당 쿠버네티스 오브젝트의 실제 상태

  - 이 두 상태가 일치하지 않을 경우,
    쿠버네티스는 **의도하는 상태(desired status)**로 바꾸기 위한 관리 시작

STATUS 예시

Deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels: ···
spec:
  replicas: 2
  selector: ···
  template: ···
```

Status

```yaml
status:
  availableReplicas: 1
  conditions:
  - lastTransitionTime: "2020-01-24T10:54:59Z"
    lastUpdateTime: "2020-01-24T10:54:59Z"
    message: Deployment has minimum availability.
    reason: MinimumReplicasAvailable
    status: "True"
    type: Available
  - lastTransitionTime: "2020-01-24T10:54:56Z"
    lastUpdateTime: "2020-01-24T10:54:59Z"
    message: ReplicaSet "nginx-deployment-7d64f4b574" has successfully progressed.
    reason: NewReplicaSetAvailable
    status: "True"
    type: Progressing
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1
  updatedReplicas: 1
```

Deployment-result.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations: ···
  creationTimestamp: "2020-01-24T10:54:56Z"
  generation: 1
  labels: ···
  name: nginx-deployment
  namespace: default
  resourceVersion: "96574"
  selfLink: /apis/apps/v1/namespaces/default/deployments/nginx-deployment
  uid: e1075fa3-6468-43d0-83c0-63fede0dae51
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector: ···
  strategy: ···
  template: ···
status:
  availableReplicas: 1
  conditions:
  - lastTransitionTime: "2020-01-24T10:54:59Z" ···
  - lastTransitionTime: "2020-01-24T10:54:56Z" ···
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1
  updatedReplicas: 1
```

STATUS 기반 K8S의 동작

Deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:···
spec:
  replicas: 2
  selector:···
  template:···
```

≠

Status

```yaml
status:
  availableReplicas: 1
  conditions:
  - lastTransitionTime: "2020-01-24T10:54:59Z"
    lastUpdateTime: "2020-01-24T10:54:59Z"
    message: Deployment has minimum availability.
    reason: MinimumReplicasAvailable
    status: "True"
    type: Available
  - lastTransitionTime: "2020-01-24T10:54:56Z"
    lastUpdateTime: "2020-01-24T10:54:59Z"
    message: ReplicaSet "nginx-deployment-7d64f4b574" has successfully progressed.
    reason: NewReplicaSetAvailable
    status: "True"
    type: Progressing
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1
  updatedReplicas: 1
```

- Compare desired status and actual status
- Kubernetes detect problem and create replica as soon as possible (1 → 2)

explain 명령어

- kubectl explain [쿠버네티스 리소스]
- 해당 쿠버네티스 리소스의 정보, 관련 field에 대한 설명 및 데이터 타입 확인 가능

- ex) kubectl explain pods

```
KIND:     Pod
VERSION:  v1

DESCRIPTION:
     Pod is a collection of containers that can run on a host. This resource is
     created by clients and scheduled onto hosts.

FIELDS:
   apiVersion   <string>
     APIVersion defines the versioned schema of this representation of an
     object. Servers should convert recognized schemas to the latest internal
     value, and may reject unrecognized values. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources

   kind <string>
     Kind is a string value representing the REST resource this object
     represents. Servers may infer this from the endpoint the client submits
     requests to. Cannot be updated. In CamelCase. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

   metadata     <Object>
     Standard object's metadata. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata

   spec <Object>
     Specification of the desired behavior of the pod. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status

   status       <Object>
     Most recently observed status of the pod. This data may not be up to date.
     Populated by the system. Read-only. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status
```

Labels & Selectors

Kubernetes 내부에서의 객체간 연계, 연결을 위한 방법

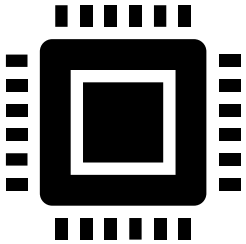
Labels & Selectors 기본정의

- A method to keep things organized, and to help you (a human) and Kuberenetes (a machine) identify resources to act upon

- Labels are key/value pairs that you can attach to objects like pods
  - They are for users to help describe meaningful and relevant information about an object
  - They do not affect the semantics of the core system

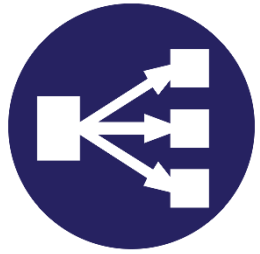
- Selectors are a way of expressing how to select objects based on their labels
  - You can specify if a label equals a given criteria or if it fits inside a set of criteria
    - Equality-based
    - Set-based

Adding Labels To Any Resources

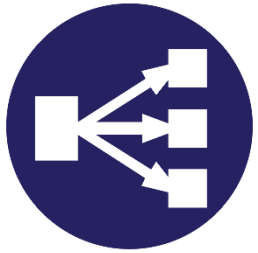


name: dcn-gateway
env: prod

name: dcn-db
env: prod

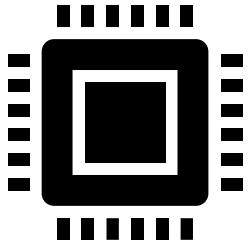


name: dcn-elb
env: prod



name: dcn-elb
env: dev

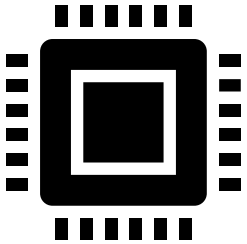
name: dcn-db
env: dev



name: dcn-gateway
env: dev

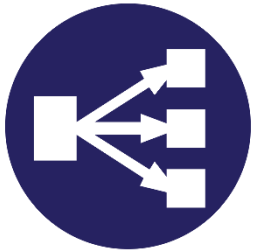
Selectors

- Selectors allows us to filter objects based on labels.
- Example:
  - 1. show me all the objects which has label where env: prod



name: dcn-gateway
env: prod

name: dcn-db
env: prod

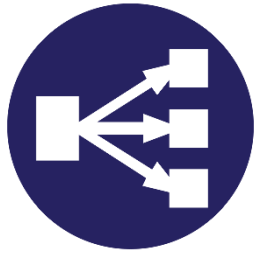


name: dcn-elb
env: prod

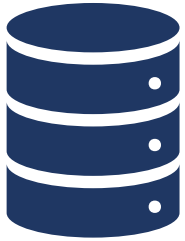
Selectors

Example:

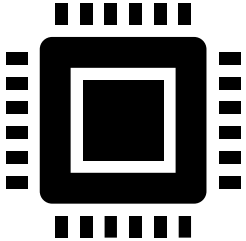
- 2. show me all the objects which has label where env: dev



name: dcn-elb
env: dev



name: dcn-db
env: dev



name: dcn-gateway
env: dev