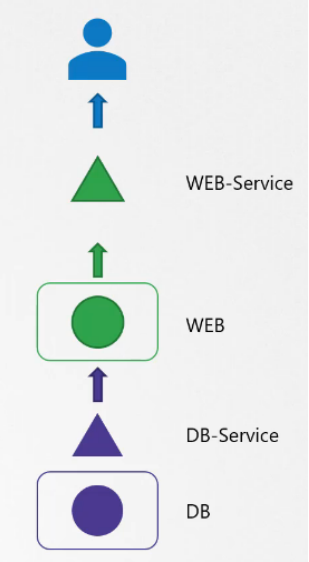
Section 15, 16: Troubleshooting, Other Topics

Section 15: Troubleshhoing

299. Application Failure



사용자들의 web application access에 문제가 있을 때

1. access 확인 : `curl http://web-service-ip:node-port`
2. service 상태 확인: `kubectl describe service web-service`
    - Endpoint 항목 확인
        - → 확인 불가 시에는 service의 Selector 항목 확인 → Pod yaml에 기재된 labels.name과 일치하는지 확인
3. pod 확인
    - pod 상태, 재시작 횟수 확인: `kubectl get pod` - STATUS, RESTARTS
    - pod의 event 확인 : `kubectl describe pod web` - Events 항목
    - pod의 log 확인: `kubectl logs web -f`
4. 의존하는 service 확인: web이 db-service에 의존하고 있기 때문에, db-service와 db pod를 확인해준다.

공식문서 링크: https://kubernetes.io/docs/tasks/debug/debug-application/

context namespace 변경: `k config set-context --current --namespace=gamma`

302. Control Plane Failure

1. 노드 상태 확인: `k get nodes`
2. pod 상태 확인: `k get pods`
3. kubeadm 으로 배포된 클러스터에 controlplane의 구성요소가 pod로 배포된 경우, `kube-system` namespace 확인: `k get pods -n kube-system`
4. controlplane의 구성요소가 service로 배포된 경우, 서비스 상태 확인:
   - 마스터 노드
     - `service kube-apiserver status`
     - `service kube-controller-manager status`
     - `service kube-scheduler status`
   - Worker Node
     - `service kubelet status`
     - `service kube-proxy status`
5. service log 확인
   - kubeadm 의 경우, controlplane 구성요소를 호스팅하는 pod의 log 확인 : `k logs kube-apiserver-master -n kube-system`
   - master node에 기본적으로 구성된 서비스의 경우, host loging solution 사용하여 서비스 로그 확인
     - ex) journal control utility 사용: `sudo journalctl -u kube-apiserver`

305. Worker Node Failure

- 노드 상태 확인: `k get nodes`
  - NotReady Status인 노드가 있으면 해당 노드 자세히 확인 `k describe node worker-1`
- 노드에서 가능한 CPU 메모리와 디스크 공간 확인: `top` , `df -h`
- kubelet 상태 확인: `service kubelet status`
- kubelet log 확인: `sudo journalctl -u kubelet`
- kubelet certificates 확인 (만료일자, ca가 올바른지 등 확인) : `openssl x509 -in /var/lib/kubelet/worker-1.crt -text`

Section 16: Other Topics

JSON PATH, Advanced Kubectl Commands

- JSON PATH를 kubetcl 명령어와 함께 사용하면 복잡한 쿠버네티스 리소스에서 필요한 정보만 정확하게 가져올 수 있음
  - 출력을 필터링하고 원하는 형태로 format할 수 있음

기본 문법

- `$.` 또는 `.`: 루트 객체에서 시작
- `..`: 재귀적 하위 요소 검색
- `*` : 와일드카드, 모든 객체/요소
- `['key']` 또는 `.key`: 특정 키에 접근
- `[index]`: 배열의 특정 인덱스 요소에 접근
  - ex) `.spec.containers[0]`
- `[start:end]`: 배열의 범위 지정
- `[?(<expression>)]`: 필터 표현식

사용법

- `kubectl get <resource> -o=jsonpath='{<expression>}'`
  - ex) `kubectl get pods -o=jsonpath='{.items.spec.containers[0].image}'`
  - ex) `kubetcl get nodes -o=jsonpath='{.items[*].metadata.name}'` : 와일드카드 이용
  - ex) `kubetcl get nodes -o=jsonpath='{.items[*].metadata.name} {"\n"} {.items[*].status.capacity.cpu}'` : 단일 명령으로 한번에 두가지 결과 얻기 (중간에 개행 추가)
- `kubectl get <resource> -o=custom-columns=<COLUMN NAME>:<JSON PATH>` : 칼럼명 custom
  - ex) `kubetcl get nodes -o=custom-columns=NODE:.metadata.name , CPU:.status.capacity.cpu` : 쉼표로 구분해서 깔끔하게 출력
- `kubectl get <resource> --sort-by='{<jsonpath-expression>}'` : 출력 결과를 특정 json path 이용해 정렬
  - ex) `kubetcl get nodes --sort-by= .metadata.name`

▼ 필터링 예시

- 이름이 redis-container인 컨테이너의 restartCount: `.status.containerStatuses[?(@.name == 'redis-container')].restartCount`

```
{
  "apiVersion": "v1",
  "kind": "Pod",
  "metadata": {
    "name": "nginx-pod",
    "namespace": "default"
  },
  "spec": {
    "containers": [
      {
        "image": "nginx:alpine",
        "name": "nginx"
      },
```

```
    {
      "image": "redis:alpine",
      "name": "redis-container"
    }
  ],
  "nodeName": "node01"
},
"status": {
  "conditions": [
    {
      "lastProbeTime": null,
      "lastTransitionTime": "2019-06-13T05:34:09Z",
      "status": "True",
      "type": "Initialized"
    },
    {
      "lastProbeTime": null,
      "lastTransitionTime": "2019-06-13T05:34:09Z",
      "status": "True",
      "type": "PodScheduled"
    }
  ],
  "containerStatuses": [
    {
      "image": "nginx:alpine",
      "name": "nginx",
      "ready": false,
      "restartCount": 4,
      "state": {
        "waiting": {
          "reason": "ContainerCreating"
        }
      }
    },
    {
      "image": "redis:alpine",
      "name": "redis-container",
      "ready": false,
      "restartCount": 2,
      "state": {
        "waiting": {
          "reason": "ContainerCreating"
        }
      }
    }
  ],
  "hostIP": "172.17.0.75",
  "phase": "Pending",
```

```
    "qosClass": "BestEffort",
    "startTime": "2019-06-13T05:34:09Z"
  }
}
```