ConfigMap

OW

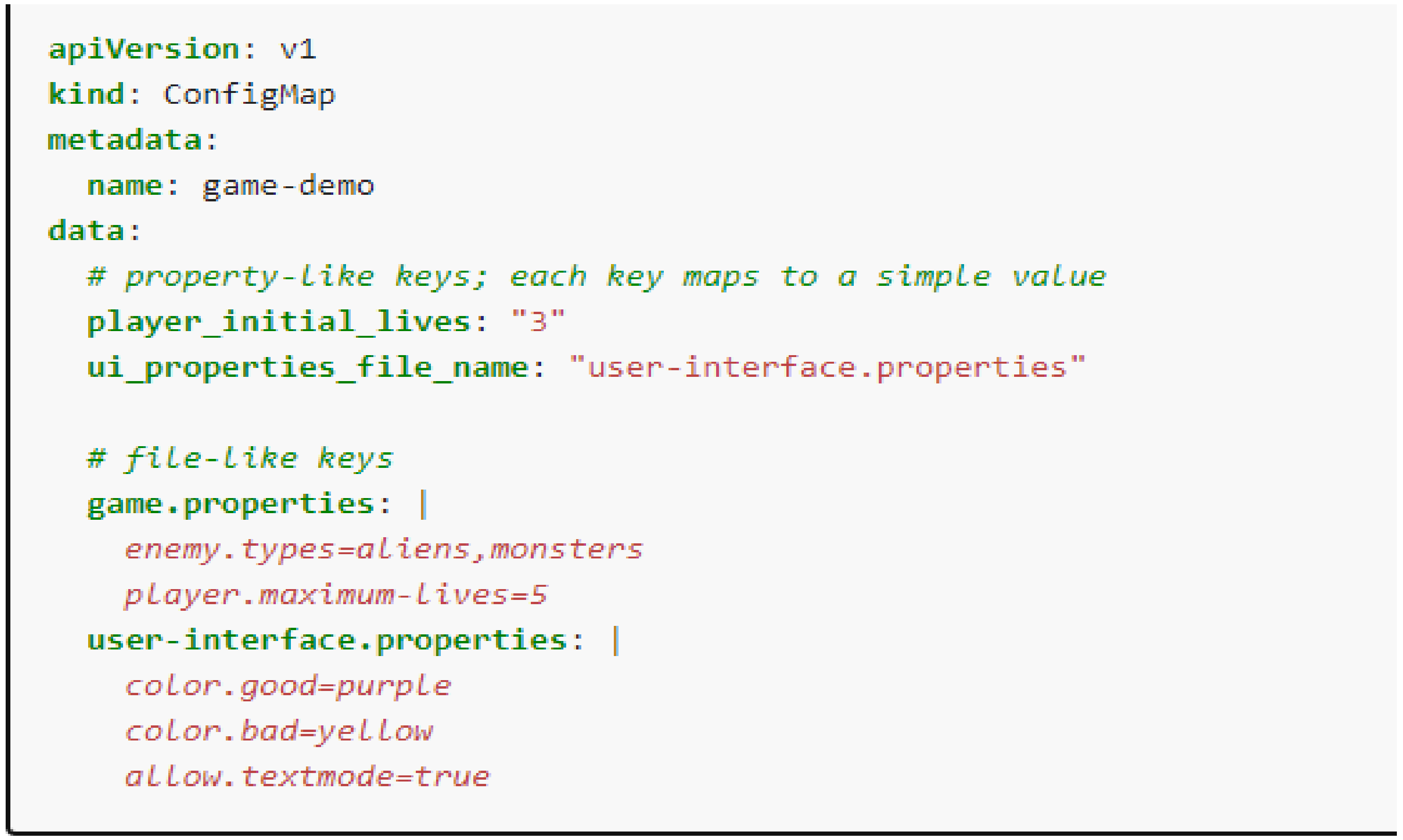
What is ConfigMaps?

- A ConfigMap is an api object to store non-confidential data in key-value format.
- Pods can consumed these config map as environment variable,command-line arguments or configuration file in a volume.
- Decouple environment specific configuration from the image.
- ConfigMap doesn’t provide encryption.
- ConfigMap cannot hold data more than 1 Mib,

ConfigMap Object

- ConfigMap allows to store data, which other objects consume.
- In spec section of ConfigMap api, we provide data or binaryData.
- data or binaryData are optional field in ConfigMap api.
- Each key under the data or the binaryData field must consist of alphanumeric characters, -, _ or .
- The Pod and the ConfigMap must be in the same namespace.
- kubelet uses the data from the ConfigMap when it launches container(s) for a Pod.

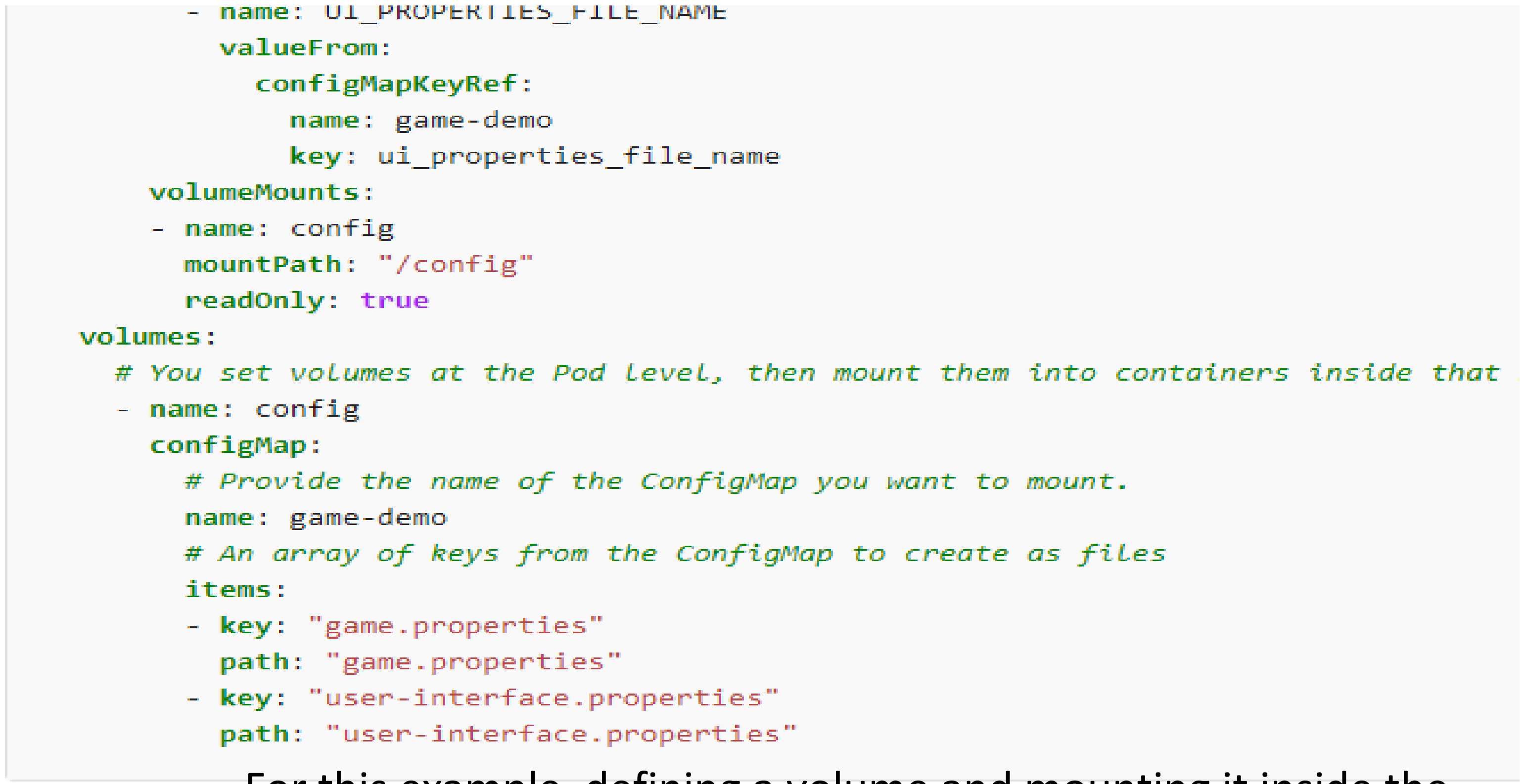
yaml



```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  # property-like keys; each key maps to a simple value
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"

  # file-like keys
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
  user-interface.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
```

Mapping in Pod



```yaml
      - name: UI_PROPERTIES_FILE_NAME
        valueFrom:
          configMapKeyRef:
            name: game-demo
            key: ui_properties_file_name
    volumeMounts:
    - name: config
      mountPath: "/config"
      readOnly: true
volumes:
  # You set volumes at the Pod level, then mount them into containers inside that
  - name: config
    configMap:
      # Provide the name of the ConfigMap you want to mount.
      name: game-demo
      # An array of keys from the ConfigMap to create as files
      items:
      - key: "game.properties"
        path: "game.properties"
      - key: "user-interface.properties"
        path: "user-interface.properties"
```

For this example, defining a volume and mounting it inside the demo container as /config creates two files, /config/game.properties and /config/user-interface.properties, even though there are four keys in the ConfigMap.

..

- Create a ConfigMap or use an existing one. Multiple Pods can reference the same ConfigMap.
- Modify your Pod definition to add a volume under .spec.volumes[]. Name the volume anything, and have a .spec.volumes[].configMap.name field set to reference your ConfigMap object.
- Add a .spec.containers[].volumeMounts[] to each container that needs the ConfigMap. Specify .spec.containers[].volumeMounts[].readOnly = true and .spec.containers[].volumeMounts[].mountPath to an unused directory name where you would like the ConfigMap to appear.
- Modify your image or command line so that the program looks for files in that directory. Each key in the ConfigMap data map becomes the filename under mountPath.