



KodeKloud

JSON PATH
FOR BEGINNERS



YAML Introduction



WHAT IS YAML?

XML

```
<Servers>
  <Server>
    <name>Server1</name>
    <owner>John</owner>
    <created>12232012</created>
    <status>active</status>
  </Server>
</Servers>
```

JSON

```
{
  Servers: [
    {
      name: Server1,
      owner: John,
      created: 12232012,
      status: active,
    }
  ]
}
```

YAML

```
Servers:
  - name: Server1
    owner: John
    created: 12232012
    status: active
```



WHAT IS YAML?

XML

```
<Servers>
  <Server>
    <name>Server1</name>
    <owner>John</owner>
    <created>12232012</created>
    <status>active</status>
  </Server>
</Servers>
```

JSON

```
{
  Servers: [
    {
      name: Server1,
      owner: John,
      created: 12232012,
      status: active,
    }
  ]
}
```

YAML

```
Servers:
  - name: Server1
    owner: John
    created: 12232012
    status: active
```



YAML - NOTES

Dictionary/Map

```
Banana:
  Calories: 105
  Fat: 0.4 g
  Carbs: 27 g
```



```
Banana:
  Calories: 105
  Carbs: 27 g
  Fat: 0.4 g
```



Dictionary – Unordered
List – Ordered

Array/List

```
Fruits:
- Orange
- Apple
- Banana
```



```
Fruits:
- Orange
- Banana
- Apple
```

```
# List of Fruits
Fruits:
- Orange
- Apple
- Banana
```



Hash # – Comments



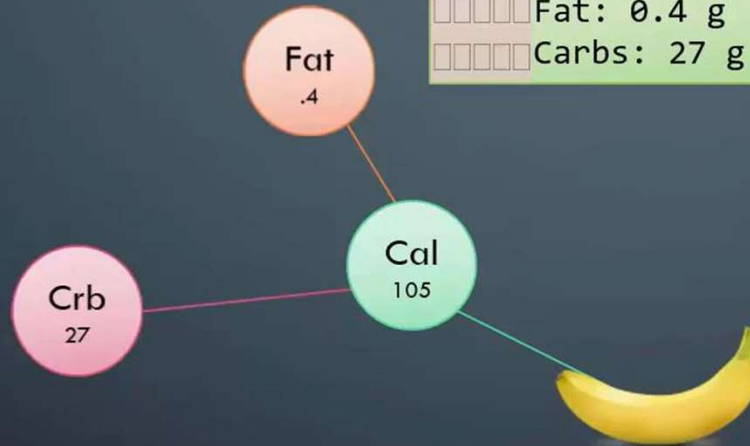
SPACES

Press `Esc` to exit full screen

Dictionary/Map

Banana:
Calories: 105
Fat: 0.4 g
Carbs: 27 g

 Equal number of spaces



YAML - ADVANCED

Press `Esc` to exit full screen

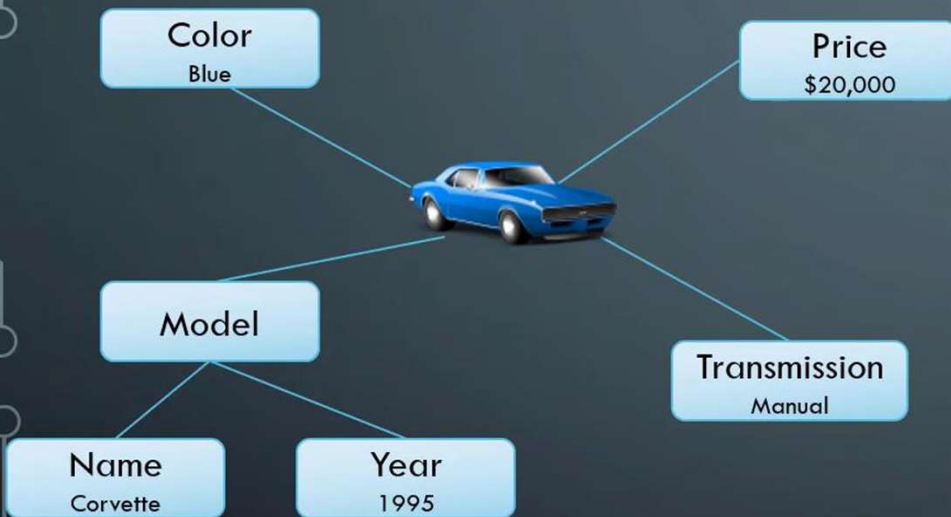
Key Value/Dictionary/Lists

Fruits:

- Banana:
 - Calories: 105
 - Fat: 0.4 g
 - Carbs: 27 g
- Grape:
 - Calories: 62
 - Fat: 0.3 g
 - Carbs: 16 g



DICTIONARY vs LIST vs LIST OF DICTIONARIES



Dictionary In Dictionary

```
Color: Blue
Model:
  Name: Corvette
  Year: 1995
Transmission: Manual
Price: $20,000
```



DICTIONARY vs LIST vs LIST OF DICTIONARIES



Color: Blue
Model:
 Name: Corvette
 Model: 1995
Transmission: Manual
Price: \$20,000



Color: Grey
Model:
 Name: Corvette
 Model: 1995
Transmission: Manual
Price: \$22,000



Color: Red
Model:
 Name: Corvette
 Model: 1995
Transmission: Automatic
Price: \$20,000



Color: Green
Model:
 Name: Corvette
 Model: 1995
Transmission: Manual
Price: \$23,000



Color: Blue
Model:
 Name: Corvette
 Model: 1995
Transmission: Manual
Price: \$20,000



Color: Black
Model:
 Name: Corvette
 Model: 1995
Transmission: Automatic
Price: \$25,000

List Of Dictionaries

- Color: Blue
Model:
 Name: Corvette
 Model: 1995
Transmission : Manual
Price: \$20,000
- Color: Grey
Model:
 Name: Corvette
 Model: 1995
Transmission: Manual
Price: \$22,000
- Color: Red
Model:
 Name: Corvette
 Model: 1995
Transmission : Automatic
Price: \$20,000
- Color: Green
Model:
 Name: Corvette
 Model: 1995
Transmission : Manual
Price: \$23,000
- Color: Blue
Model:
 Name: Corvette
 Model: 1995
Transmission : Manual
Price: \$20,000

YAML - NOTES

Dictionary/Map

Banana:

Calories: 105

Fat: 0.4 g

Carbs: 27 g

=

Banana:

Calories: 105

Carbs: 27 g

Fat: 0.4 g



Dictionary – Unordered
List – Ordered

Array/List

Fruits:

- Orange
- Apple
- Banana

≠

Fruits:

- Orange
- Banana
- Apple

List of Fruits

Fruits:

- Orange
- Apple
- Banana



Hash # – Comments





KodeKloud

Check out our full JSON Path course here: <https://kode.wiki/3NuVhVV>

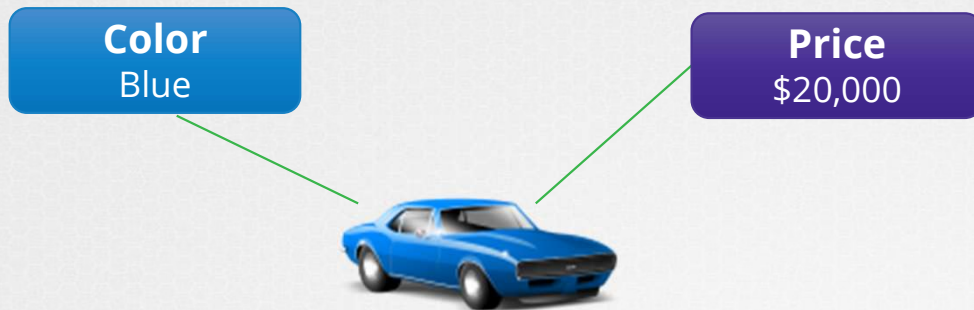
JSON PATH

PART – 1 - INTRODUCTION

Objectives

- YAML
- YAML vs JSON
- JSON PATH
 - Dictionaries
 - Lists
 - Lists and Dictionaries
 - Criteria
- Practice Exercises

YAML



```
car:  
  color: blue  
  price: $20,000
```


YAML vs JSON

```
{  
  "car": {  
    "color": "blue",  
    "price": "$20,000"  
  }  
}
```

```
car:  
  color: blue  
  price: $20,000
```

YAML vs JSON

```
{
  "car": {
    "color": "blue",
    "price": "$20,000",
    "wheels": [
      {
        "model": "X345ERT",
        "location": "front-right"
      },
      {
        "model": "X345ERT",
        "location": "front-left"
      },
      {
        "model": "X345ERT",
        "location": "rear-right"
      },
      {
        "model": "X345ERT",
        "location": "rear-right"
      }
    ]
  }
}
```

```
car:
  color: blue
  price: $20,000
  wheels:
    - model: X345ERT
      location: front-right
    - model: X345ERT
      location: front-left
    - model: X345ERT
      location: rear-right
    - model: X345ERT
      location: rear-right
```

Convert JSON to YAML

https://www.json2yaml.com

JSON

Copy JSON

```
1 {
2   "car": {
3     "color": "blue",
4     "price": "$20,000",
5     "wheels": [
6       {
7         "model": "X345ERT",
8         "location": "front-right"
9       },
10      {
11        "model": "X345ERT",
12        "location": "front-left"
13      },
14      {
15        "model": "X345ERT",
16        "location": "rear-right"
17      },
18      {
19        "model": "X345ERT",
20        "location": "rear-right"
21      }
22    ]
23  }
24 }
```

YAML

Copy YAML

```
1 ---
2 car:
3   color: blue
4   price: "$20,000"
5   wheels:
6     - model: X345ERT
7       location: front-right
8     - model: X345ERT
9       location: front-left
10    - model: X345ERT
11      location: rear-right
12    - model: X345ERT
13      location: rear-right
14
```

JSON PATH

```
{  
  "car": {  
    "color": "blue",  
    "price": "$20,000"  
  }  
}
```

```
car:  
  color: blue  
  price: $20,000
```

Query

DAT A

Car	Color	Price	Year
1	Blue	\$20,000	1987
2	Red	\$22,000	1988
3	Yellow	\$18,000	1989

1

Get color and price of cars

```
select color,price from cars;
```

2

Get blue car details

```
select * from cars  
where color is "Blue";
```

3

Get price of the blue car

```
select price from cars  
where color is "Blue";
```

RESULT

Color	Price
Blue	\$20,000
Red	\$22,000
Yellow	\$18,000

Car	Color	Price	Year
1	Blue	\$20,000	1987

Price

\$20,000

JSON PATH

**DAT
A**

```
{  
  "car": {  
    "color": "blue",  
    "price": "$20,000"  
  }  
}
```

QUERY

```
Get car details  
car
```

RESULT

```
{  
  "color": "blue",  
  "price": "$20,000"  
}
```


JSON PATH - Dictionaries

DAT A

```
{
  "car": {
    "color": "blue",
    "price": "$20,000"
  },
  "bus": {
    "color": "white",
    "price": "$120,000"
  }
}
```

QUERY

```
Get car details
car
```

```
Get bus details
bus
```

```
Get car's color
car.color
```

```
Get bus's color
bus.price
```

RESULT

```
{
  "color": "blue",
  "price": "$20,000"
}
```

```
{
  "color": "white",
  "price": "$120,000"
}
```

```
"blue"
```

```
"$120,000"
```

JSON PATH - Dictionaries

DAT A

```
{
  "vehicles": {
    "car": {
      "color": "blue",
      "price": "$20,000"
    },
    "bus": {
      "color": "white",
      "price": "$120,000"
    }
  }
}
```

QUERY

Get car details

vehicles.car

Get bus details

vehicles.bus

Get car's color

vehicles.car.color

Get bus's color

vehicles.bus.price

RESULT

```
{
  "color": "blue",
  "price": "$20,000"
}
```

```
{
  "color": "white",
  "price": "$120,000"
}
```

```
"blue"
```

```
"$120,000"
```

Root element

QUERY

RESULT

```
{  
  "car": {  
    "color": "blue",  
    "price": "$20,000"  
  },  
  "bus": {  
    "color": "white",  
    "price": "$120,000"  
  }  
}
```

Get car details
vehicles\$.car

```
{  
  "color": "blue",  
  "price": "$20,000"  
}
```

Get bus details
vehicles\$.bus

```
{  
  "color": "white",  
  "price": "$120,000"  
}
```

Get car's color
vehicles\$.car.color

```
"blue"
```

Get bus's color
vehicles\$.bus.price

```
"$120,000"
```

JSON PATH - Dictionaries

DAT A

```
{
  "vehicles": {
    "car": {
      "color": "blue",
      "price": "$20,000"
    },
    "bus": {
      "color": "white",
      "price": "$120,000"
    }
  }
}
```

QUERY

Get car details

`$.vehicles.car`

Get bus details

`$.vehicles.bus`

Get car's color

`$.vehicles.car.color`

Get bus's color

`$.vehicles.bus.price`

RESULT

```
[
  {
    "color": "blue",
    "price": "$20,000"
  }
]
```

```
[
  {
    "color": "white",
    "price": "$120,000"
  }
]
```

```
[
  "blue"
]
```

```
[
  "$120,000"
]
```

JSON PATH - Lists

DAT A

```
[  
0 "car",  
1 "bus",  
2 "truck",  
3 "bike"  
]
```

QUERY

Get the 1st element

```
$.[0]
```

Get the 4th element

```
$.[3]
```

Get the 1st and 4th element

```
$.[0,3]
```

RESULT

```
[ "car" ]
```

```
[ "bike" ]
```

```
[ "car", "bike" ]
```


JSON PATH – Dictionary & Lists

DAT

A

```
{
  "car": {
    "color": "blue",
    "price": "$20,000",
    "wheels": [
      {
        "model": "X345ERT",
        "location": "front-right"
      },
      {
        "model": "X346GRX",
        "location": "front-left"
      },
      {
        "model": "X236DEM",
        "location": "rear-right"
      },
      {
        "model": "X987XMV",
        "location": "rear-right"
      }
    ]
  }
}
```

QUERY

```
Get the model of the 2nd wheel
$.car.wheels[1].model
```

RESULT

```
[
  {
    "model": "X345ERT",
    "location": "front-right"
  },
  {
    "model": "X346GRX",
    "location": "front-left"
  },
  {
    "model": "X236DEM",
    "location": "rear-right"
  },
  {
    "model": "X987XMV",
    "location": "rear-right"
  }
]
```


JSON PATH – Criteria

DAT A

```
[
  12,
  43,
  23,
  12,
  56,
  43,
  93,
  32,
  45,
  63,
  27,
  8,
  78
]
```

QUERY

Get all numbers greater than 40

```
$[ Check if each item in the array > 40 ]
```

Check if => ? ()

```
$[?( each item in the list > 40 )]
```

each item in the list => @

```
$[?( @ > 40 )]
```

@ == 40

@ in [40,43,45]

@ != 40

@ nin [40,43,45]

RESULT

```
[
  43,
  56,
  43,
  93,
  45,
  63,
  78
]
```

JSON PATH – Criteria

DAT

A

```
{
  "car": {
    "color": "blue",
    "price": "$20,000",
    "wheels": [
      {
        "model": "X345ERT",
        "location": "front-right"
      },
      {
        "model": "X346GRX",
        "location": "front-left"
      },
      {
        "model": "X236DEM",
        "location": "rear-right"
      },
      {
        "model": "X987XMV",
        "location": "rear-left"
      }
    ]
  }
}
```

QUERY

Get the model of the rear-right wheel

`$.car.wheels[2].model`

RESULT

`"X236DEM"`

JSON PATH – Criteria

DAT

A

```
{
  "car": {
    "color": "blue",
    "price": "$20,000",
    "wheels": [
      {
        "model": "X345ERT",
        "location": "front-right"
      },
      {
        "model": "X236DEM",
        "location": "rear-right"
      },
      {
        "model": "X346GRX",
        "location": "front-left"
      },
      {
        "model": "X987XMV",
        "location": "rear-left"
      }
    ]
  }
}
```

QUERY

Get the model of the rear-right wheel

```
$.car.wheels[2].model
```

```
$.car.wheels[?(@.location == "rear-right")].model
```

RESULT

```
"X236DEM"
```

kodekloud.com/p/json-path-quiz

Dictionary

Develop a JSON path query to extract the expected output from the Source Data.

SHOW SOLUTION

Source Data:

```
1 {  
2   "property1": "value1",  
3   "property2": "value2"  
4 }
```

Expected Output:

```
1 "value1"
```

Output based on your JSON Path

✖ 1 No match

| References

<https://github.com/json-path/JsonPath>



KodeKloud

Check out our full JSON Path course here: <https://kode.wiki/3NuVhVV>

JSON PATH

PART – 2 – WILD CARD

JSON PATH – Wildcard

DAT A

```
{
  "car": {
    "color": "blue",
    "price": "$20,000"
  },
  "bus": {
    "color": "white",
    "price": "$120,000"
  }
}
```

QUERY

Get car's color
\$.car.color

Get bus's color
\$.bus.color

Get all colors
\$.*.color

Get all prices
\$.*.price

RESULT

```
[ "blue" ]
```

```
[ "white" ]
```

```
[ "blue", "white" ]
```

```
[ "$20,000", "$120,000" ]
```

JSON PATH – Wildcard

DAT A

```
[
  {
    "model": "X345ERT",
    "location": "front-right"
  },
  {
    "model": "X346ERT",
    "location": "front-left"
  },
  {
    "model": "X347ERT",
    "location": "rear-right"
  },
  {
    "model": "X348ERT",
    "location": "rear-right"
  }
]
```

QUERY

Get 1st wheel's model

`$.model`

Get 4th wheel's model

`$.model`

Get all wheels' model

`$.model`

RESULT

`["X345ERT"]`

`["X348ERT"]`

`["X345ERT", "X346ERT",
"X347ERT", "X348ERT"]`

JSON PATH – Wildcard

DAT

```
{
  "car": {
    "color": "blue",
    "price": 20000,
    "wheels": [
      {
        "model": "X345ERT",
      },
      {
        "model": "X346ERT",
      }
    ]
  },
  "bus": {
    "color": "white",
    "price": 120000,
    "wheels": [
      {
        "model": "Z227KLJ",
      },
      {
        "model": "Z226KLJ",
      }
    ]
  }
}
```

QUERY

Get car's 1st wheel model

```
$.car.wheels[0].model
```

Get car's all wheel model

```
$.car.wheels[*].model
```

Get bus's wheel models

```
$.bus.wheels[*].model
```

Get all wheels' models

```
$.*.wheels[*].model
```

RESULT

```
[ "X345ERT" ]
```

```
[ "X345ERT", "X346ERT" ]
```

```
[ "Z227KLJ", "Z226KLJ" ]
```

```
[ "X345ERT", "X346ERT",
  "Z227KLJ", "Z226KLJ" ]
```



KodeKloud

Check out our full JSON Path course here: <https://kode.wiki/3NuVhVV>

JSON PATH

PART – 3 – LISTS

JSON PATH - Lists

DAT

```
[  
  "Apple",  
  "Google",  
  "Microsoft",  
  "Amazon",  
  "Facebook",  
  "Coca-Cola",  
  "Samsung",  
  "Disney",  
  "Toyota",  
  "McDonald's"  
]
```

QUERY

Get the 1st element

`$.[0]`

Get the 4th element

`$.[3]`

Get the 1st and 4th element

`$.[0,3]`

Get the 1st to 4th element

`$.[0:3]`

START : END

`$.[0:4]`

RESULT

```
[ "Apple" ]
```

```
[ "Amazon" ]
```

```
[ "Apple", "Amazon" ]
```

```
[  
  "Apple",  
  "Google",  
  "Microsoft"  
]
```

```
[  
  "Apple",  
  "Google",  
  "Microsoft",  
  "Amazon"  
]
```

JSON PATH - Lists

DAT

```
[  
  "Apple",  
  "Google",  
  "Microsoft",  
  "Amazon",  
  "Facebook",  
  "Coca-Cola",  
  "Samsung",  
  "Disney",  
  "Toyota",  
  "McDonald's"  
]
```

QUERY

```
$(0:8]
```

START : END

```
$(0:8:2]
```

START : END : STEP

RESULT

```
[  
  "Apple",  
  "Google",  
  "Microsoft",  
  "Amazon",  
  "Facebook",  
  "Coca-Cola",  
  "Samsung",  
  "Disney"  
]
```

```
[  
  "Apple",  
  "Microsoft",  
  "Facebook",  
  "Samsung"  
]
```

JSON PATH - Lists

DAT

```
[
  0 "Apple",      -10
  1 "Google",     -9
  2 "Microsoft",  -8
  3 "Amazon",     -7
  4 "Facebook",   -6
  5 "Coca-Cola",  -5
  6 "Samsung",    -4
  7 "Disney",     -3
  8 "Toyota",     -2
  9 "McDonald's" -1
]
```

```
[
  "Apple",
  "Microsoft",
  "Facebook",
  "Samsung"
]
```

QUERY

Get the last element

`$[9]`



Get the last element

`$[-1]`



`$[-1:0]`



`$[-1:]`



Get the last 3 elements

`$[-3:]`

RESULT

```
[
  "McDonald's"
]
```

Does not work in certain implementations

```
[
  "Disney",
  "Toyota",
  "McDonald's"
]
```



KodeKloud

Check out our full JSON Path course here: <https://kode.wiki/3NuVhVV>

JSON PATH IN KUBERNETES

Objectives

- JSON PATH in KubeCtl
- Why JSON PATH?
- View and interpret KubeCtl output in JSON Format
- How to use JSON PATH with KubeCtl
- JSON PATH Examples
- Loops – Range
- Custom Columns
- Sort
- Practice Tests and Exercises

I Pre-Requisite

- JSON PATH for Beginners on Youtube
- JSON PATH Practice Tests on KodeKloud
- JSON PATH Practice Tests on Kubernetes data set on KodeKloud

www.kodekloud.com/p/json-path-quiz

Why JSON PATH?

- Large Data sets
 - 100s of Nodes
 - 1000s of PODs, Deployments, ReplicaSets

KubeCtl

```
▶ kubectl get nodes
```

▶
Kubectl



kube-
apiserver

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	master	40m	v1.11.3
node01	Ready	<none>	40m	v1.11.3

```
{
  "apiVersion": "v1",
  "items": [
    {
      "apiVersion": "v1",
      "kind": "Node",
      "metadata": {
        "annotations": {
          "node.alpha.kubernetes.io/ttl": "0",
        },
        "creationTimestamp": "2019-06-21T09:01:56Z",
        "labels": {
          "beta.kubernetes.io/arch": "amd64",
          "beta.kubernetes.io/os": "linux",
          "kubernetes.io/hostname": "master",
          "node-role.kubernetes.io/master": ""
        }
      }
    }
  ]
}
```


KubeCtl

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	master	40m	v1.11.3
node01	Ready	<none>	40m	v1.11.3

```
▶ kubectl get nodes -o wide
```

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	K
master	Ready	master	7m	v1.11.3	172.17.0.44	<none>	Ubuntu 16.04.2 LTS	4
node01	Ready	<none>	6m	v1.11.3	172.17.0.63	<none>	Ubuntu 16.04.2 LTS	4

```
{
  "apiVersion": "v1",
  "items": [
    {
      "apiVersion": "v1",
      "kind": "Node",
      "metadata": {
        "annotations": {
          "node.alpha.kubernetes.io/ttl": "0",
        },
        "creationTimestamp": "2019-06-21T09:01:56Z",
        "labels": {
          "beta.kubernetes.io/arch": "amd64",
          "beta.kubernetes.io/os": "linux",
          "kubernetes.io/hostname": "master",
          "node-role.kubernetes.io/master": ""
        }
      }
    }
  ]
}
```

KubeCtl - JSON PATH

NAME	CPU
master	4
node01	4

NAME	TAINTS
master	node-role.kubernetes.io/master
node01	

NAME	ARCHITECTURE
master	amd64
node01	amd64

NAME	IMAGE
red	nginx
blue	ubuntu
yellow	redis

How to JSON PATH in KubeCtl?

- 1 Identify the **kubectl** command
- 2 Familiarize with **JSON** output
- 3 Form the **JSON PATH** query
`.items[0].spec.containers[0].image`
- 4 Use the **JSON PATH** query with **kubectl** command

```
▶ kubectl get nodes -o json
```

```
▶ kubectl get pods -o json
```

```
{
  "apiVersion": "v1",
  "kind": "List",
  "items": [
    {
      "apiVersion": "v1",
      "kind": "Pod",
      "metadata": {
        "name": "nginx-5557945897-gznjp",
      },
      "spec": {
        "containers": [
          {
            "image": "nginx:alpine",
            "name": "nginx"
          }
        ],
        "nodeName": "node01"
      }
    }
  ]
}
```

```
▶ kubectl get pods -o=jsonpath='{
```

JSON PATH Examples

```
▶ kubectl get nodes -o=jsonpath='{.items[*].metadata.name}'
```

```
master node01
```

```
▶ kubectl get nodes -o=jsonpath='{.items[*].status.nodeInfo.architecture}'
```

```
amd64 amd64
```

```
▶ kubectl get pods -o=jsonpath='{.items[*].status.capacity.cpu}'
```

```
4 4
```

```
{
  "apiVersion": "v1",
  "items": [
    {
      "apiVersion": "v1",
      "kind": "Node",
      "metadata": {
        "name": "master"
      },
      "status": {
        "capacity": {
          "cpu": "4"
        },
        "nodeInfo": {
          "architecture": "amd64",
          "operatingSystem": "linux",
        }
      }
    },
    {
      "apiVersion": "v1",
      "kind": "Node",
      "metadata": {
        "name": "node01",
      },
      "status": {
        "capacity": {
          "cpu": "4",
        },
        "nodeInfo": {
          "architecture": "amd64",
          "operatingSystem": "linux",
        }
      }
    }
  ],
  "kind": "List",
}
```

JSON PATH Examples

```
▶ kubectl get nodes -o=jsonpath='{.items[*].metadata.name}'
```

```
master node01
```

```
▶ kubectl get nodes -o=jsonpath='{.items[*].status.nodeInfo.architecture}'
```

```
amd64 amd64
```

```
▶ kubectl get nodes -o=jsonpath='{.items[*].status.capacity.cpu}'
```

```
4 4
```

```
▶ kubectl get nodes -o=jsonpath='{.items[*].metadata.name}'
```

```
master node01 4 4
```

```
▶ kubectl get nodes -o=jsonpath='{.items[*].metadata.name}{.items[*].status.capacity.cpu}'
```

```
master node01
```

```
4 4
```

```
{"\n"}
```

New line

```
{"\t"}
```

Tab

Loops - Range

```
▶ kubectl get nodes -o=jsonpath='{.items[*].metadata.name}{"\n"}{.items[*].status.capacity.cpu}'
```

```
master node01
4         4
```

```
master      4
node01      4
```

```
FOR EACH NODE
    PRINT NODE NAME \t PRINT CPU COUNT \n
END FOR
```

```
'{range .items[*]}
    {.metadata.name} {"\t"} {.status.capacity.cpu} {"\n"}
{end}'
```

Loops - Range

```
▶ kubectl get pods -o=jsonpath='{.items[*].metadata.name} {"\n"}{.items[*].status.capacity.cpu}'
```

```
▶ kubectl get nodes -o=jsonpath='{range .items[*]}{.metadata.name} {"\t"} {.status.capacity.cpu} {"\n"}{end}'
```

JSON PATH for Custom Columns

```
▶ kubectl get nodes -o=jsonpath='{.items[*].metadata.name}{"\n"}{.items[*].status.capacity.cpu}'
```

```
master node01
4      4
```

NODE	CPU
master	4
node01	4

```
kubectl get nodes -o=custom-columns=<COLUMN NAME>:<JSON PATH>
```

```
▶ kubectl get nodes -o=custom-columns=NODE:.metadata.name ,CPU:.status.capacity.cpu
```

NODE	CPU
master	4
node01	4

JSON PATH for Sort

```
▶ kubectl get nodes -o=custom-columns=NODE:.metadata.name,CPU:.status.capacity.cpu
```

NODE	CPU
master	4
node01	4

```
▶ kubectl get nodes --sort-by=
```

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	master	5m	v1.11.3
node01	Ready	<none>	5m	v1.11.3

```
▶ kubectl get nodes --sort-by=
```

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	master	5m	v1.11.3
node01	Ready	<none>	5m	v1.11.3



KodeKloud

Check out our full JSON Path course here: <https://kode.wiki/3NuVhVV>