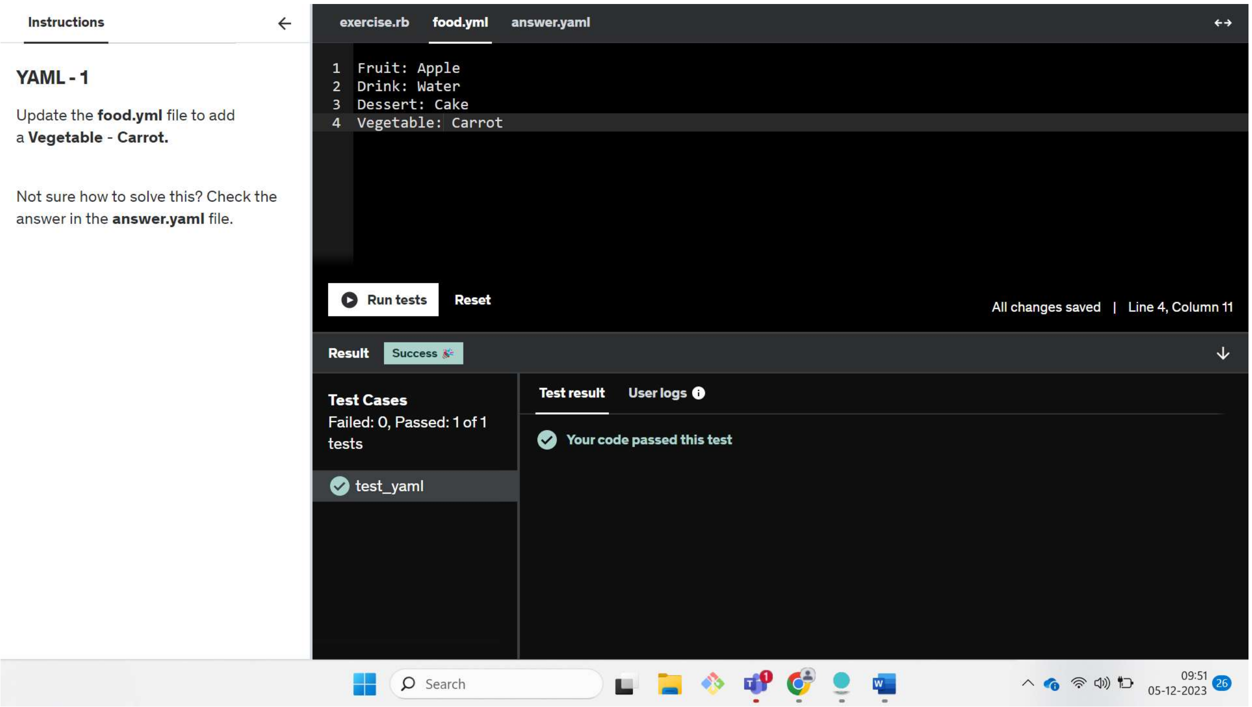
Kubernetes exercise:



YAML - 1

Update the **food.yml** file to add a **Vegetable** - **Carrot.**

Not sure how to solve this? Check the answer in the **answer.yaml** file.

```
Fruit: Apple
Drink: Water
Dessert: Cake
Vegetable: Carrot
```

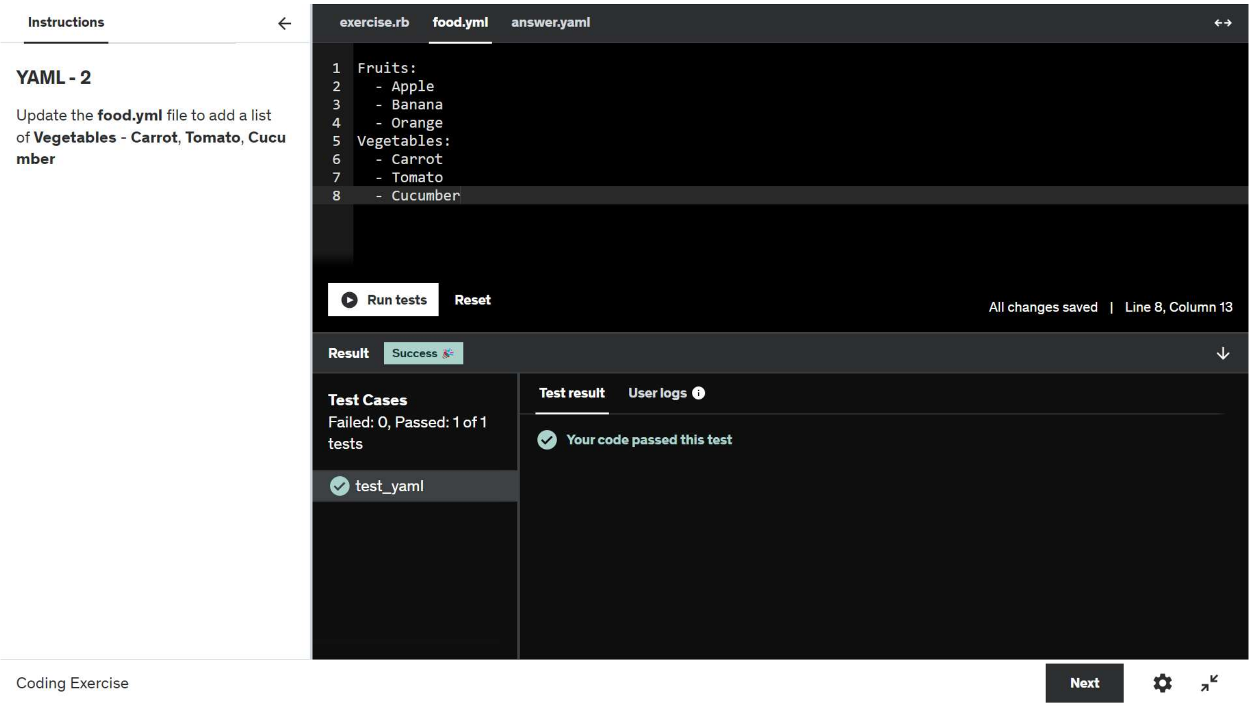
All changes saved | Line 4, Column 11

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test

09:51
05-12-2023



YAML - 2

Update the **food.yml** file to add a list of **Vegetables** - **Carrot**, **Tomato**, **Cucumber**

```
Fruits:
  - Apple
  - Banana
  - Orange
Vegetables:
  - Carrot
  - Tomato
  - Cucumber
```

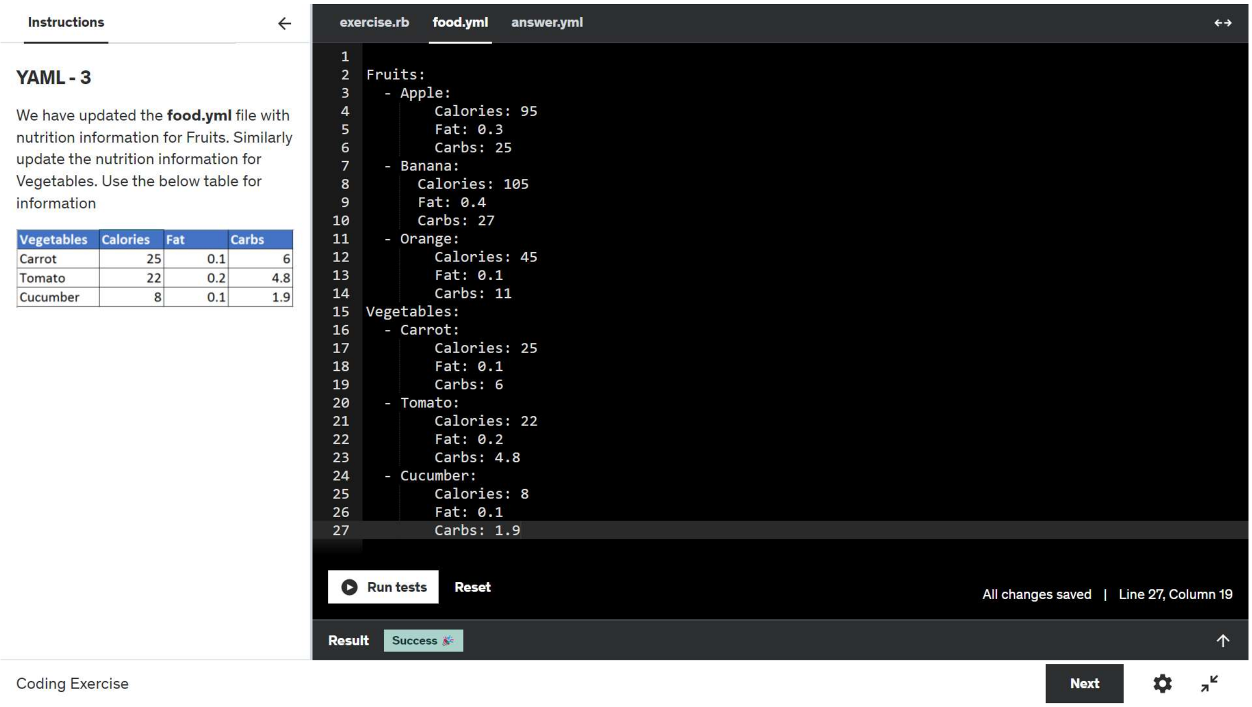
All changes saved | Line 8, Column 13

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test

Coding Exercise



YAML - 3

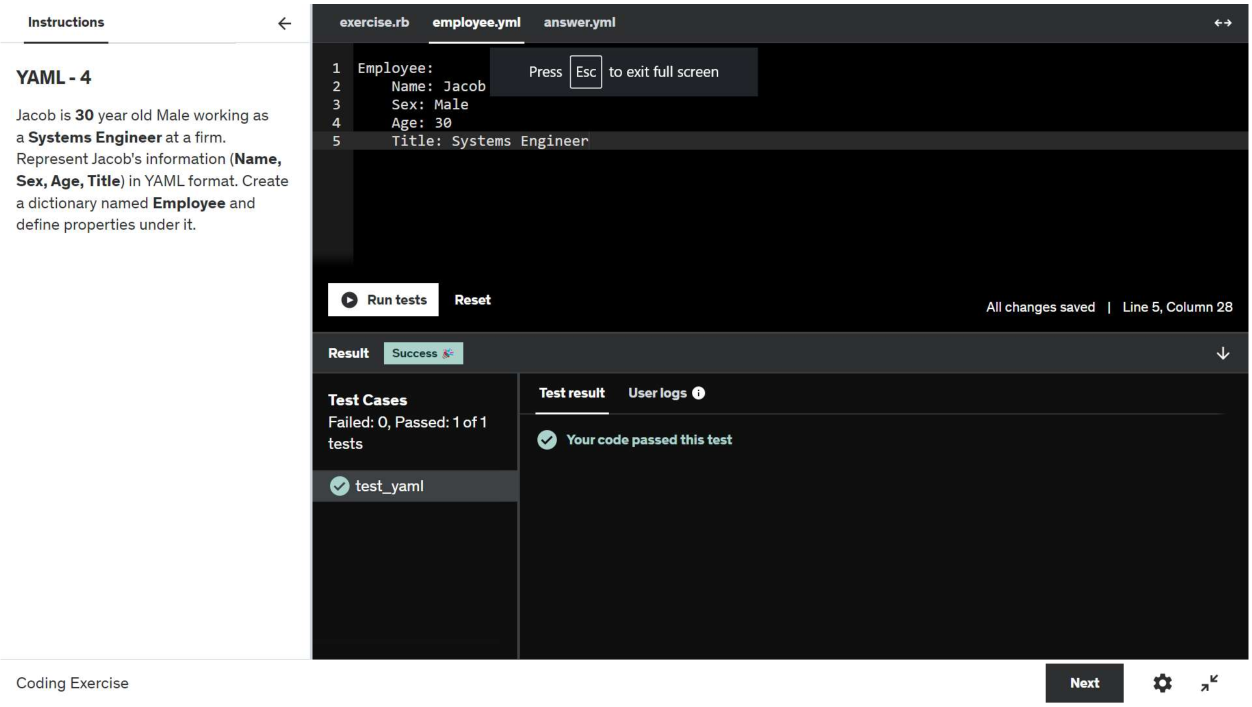
We have updated the **food.yml** file with nutrition information for Fruits. Similarly update the nutrition information for Vegetables. Use the below table for information

Vegetables	Calories	Fat	Carbs
Carrot	25	0.1	6
Tomato	22	0.2	4.8
Cucumber	8	0.1	1.9

```

Fruits:
  - Apple:
        Calories: 95
        Fat: 0.3
        Carbs: 25
  - Banana:
      Calories: 105
      Fat: 0.4
      Carbs: 27
  - Orange:
        Calories: 45
        Fat: 0.1
        Carbs: 11
Vegetables:
  - Carrot:
        Calories: 25
        Fat: 0.1
        Carbs: 6
  - Tomato:
        Calories: 22
        Fat: 0.2
        Carbs: 4.8
  - Cucumber:
        Calories: 8
        Fat: 0.1
        Carbs: 1.9
```

All changes saved | Line 27, Column 19

Coding Exercise



YAML - 4

Jacob is **30** year old Male working as a **Systems Engineer** at a firm. Represent Jacob's information (**Name, Sex, Age, Title**) in YAML format. Create a dictionary named **Employee** and define properties under it.

```
Employee:
    Name: Jacob
    Sex: Male
    Age: 30
    Title: Systems Engineer
```

Press Esc to exit full screen

All changes saved | Line 5, Column 28

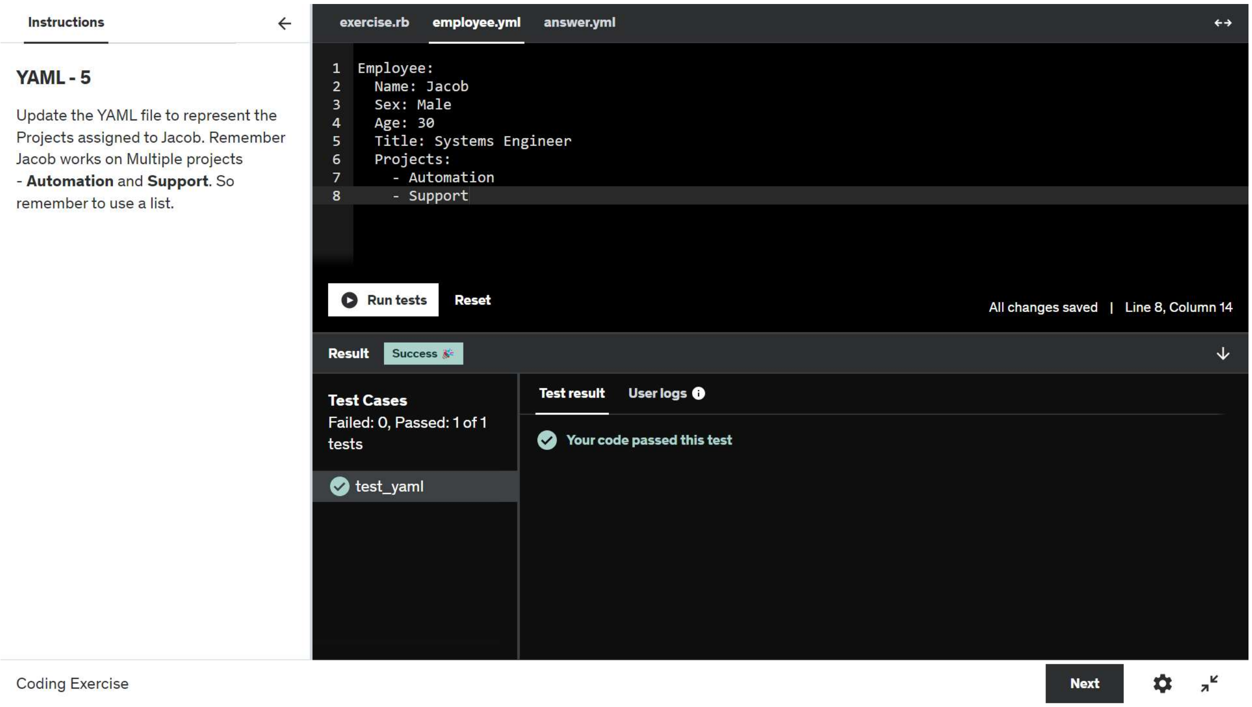
Test Cases

Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test

Coding Exercise



YAML - 5

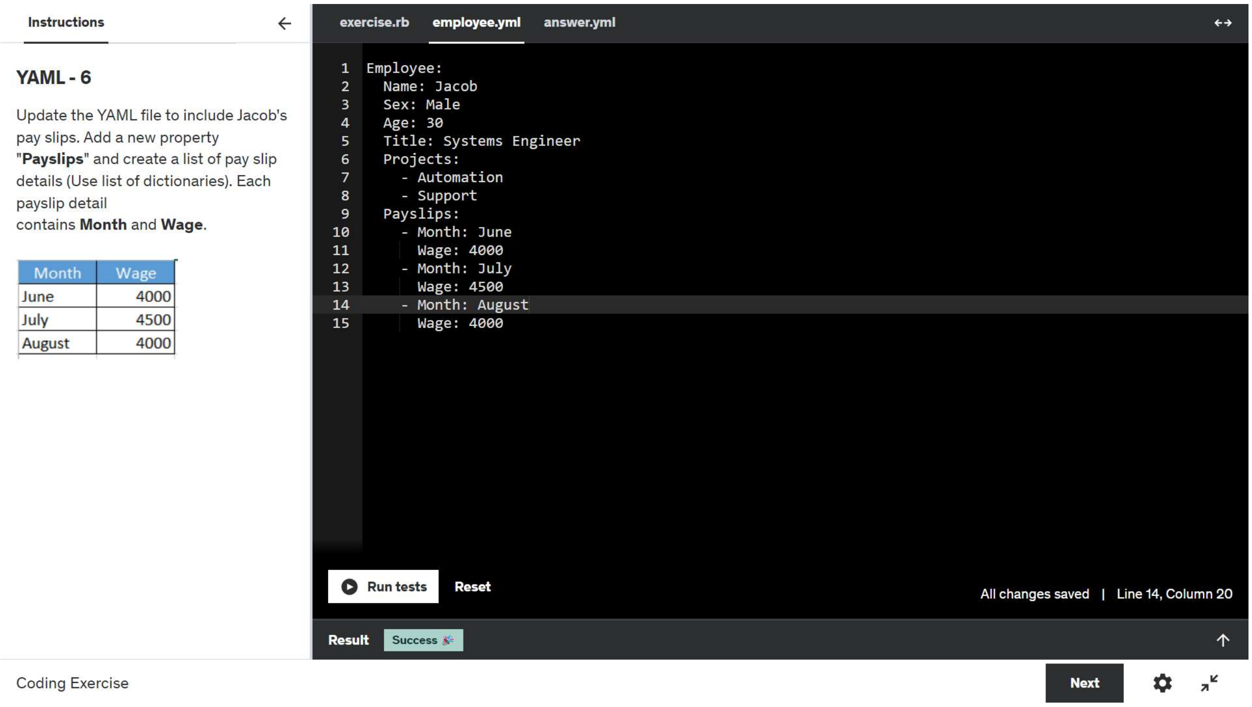
Update the YAML file to represent the Projects assigned to Jacob. Remember Jacob works on Multiple projects - **Automation** and **Support**. So remember to use a list.

```
Employee:
  Name: Jacob
  Sex: Male
  Age: 30
  Title: Systems Engineer
  Projects:
    - Automation
    - Support
```

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test

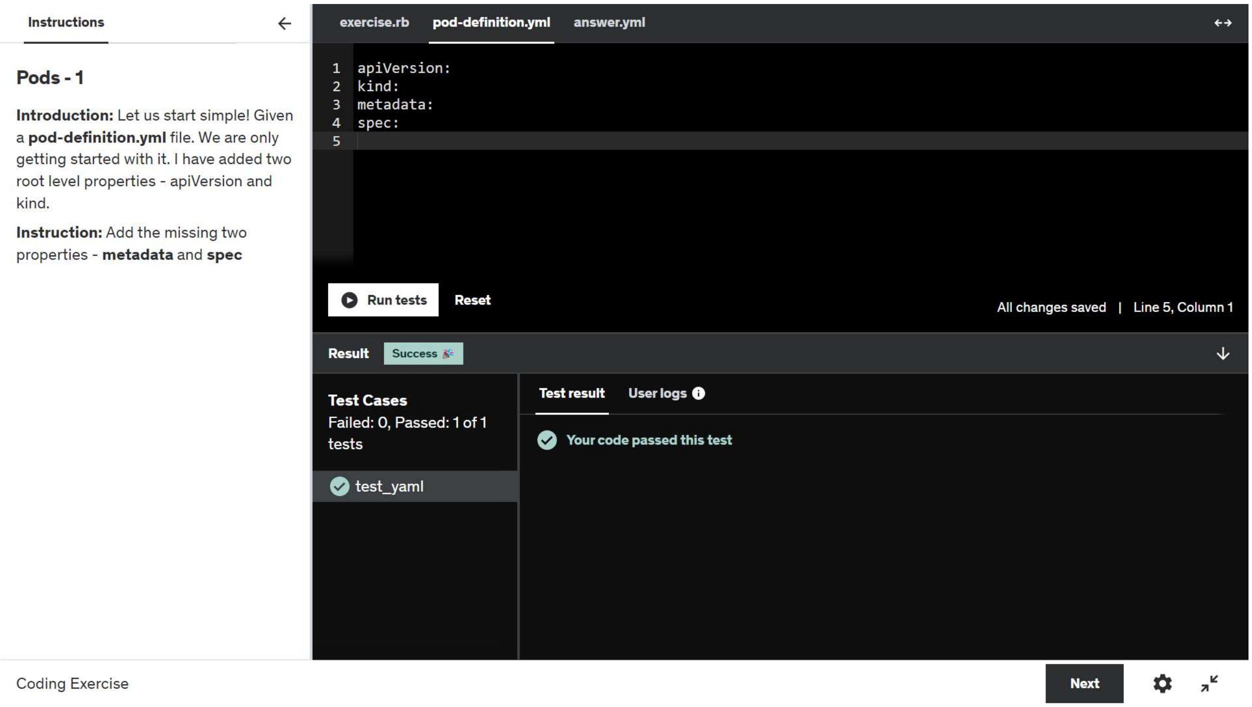


YAML - 6

Update the YAML file to include Jacob's pay slips. Add a new property "**Payslips**" and create a list of pay slip details (Use list of dictionaries). Each payslip detail contains **Month** and **Wage**.

Month	Wage
June	4000
July	4500
August	4000

```
Employee:
  Name: Jacob
  Sex: Male
  Age: 30
  Title: Systems Engineer
  Projects:
    - Automation
    - Support
  Payslips:
    - Month: June
      Wage: 4000
    - Month: July
      Wage: 4500
    - Month: August
      Wage: 4000
```



Pods - 1

Introduction: Let us start simple! Given a **pod-definition.yml** file. We are only getting started with it. I have added two root level properties - apiVersion and kind.

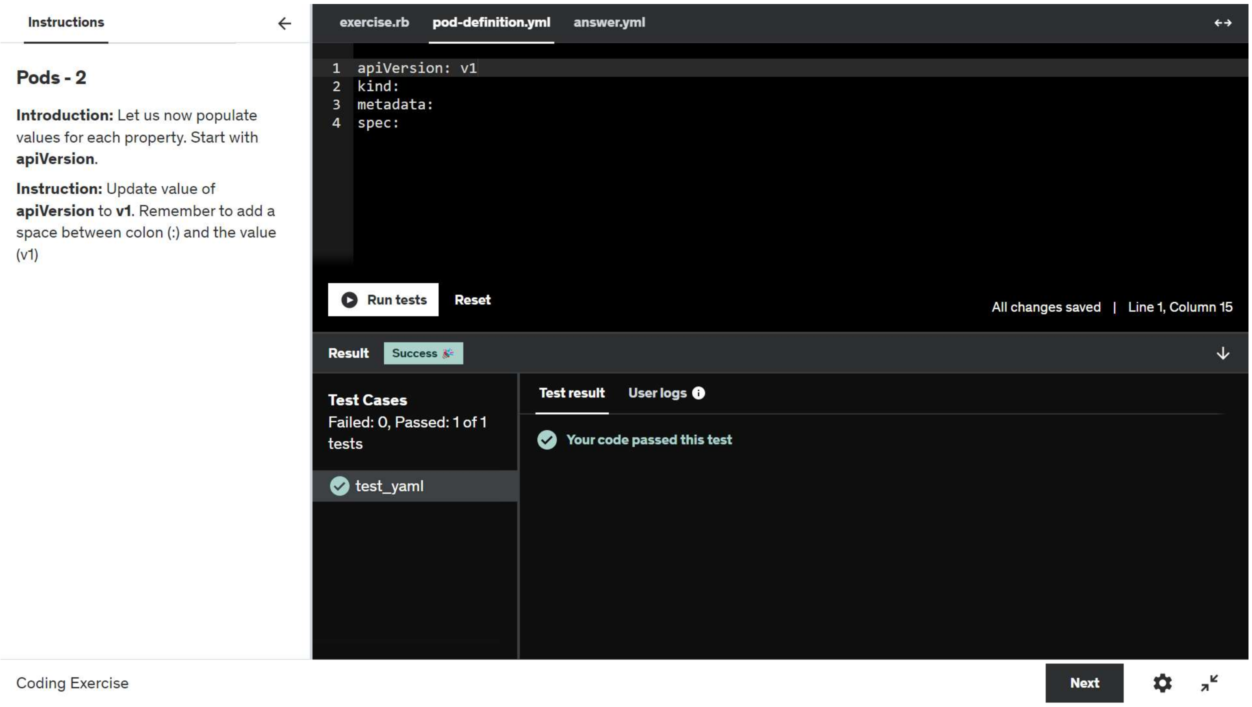
Instruction: Add the missing two properties - **metadata** and **spec**

```
apiVersion:
kind:
metadata:
spec:

```

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test



Pods - 2

Introduction: Let us now populate values for each property. Start with **apiVersion**.

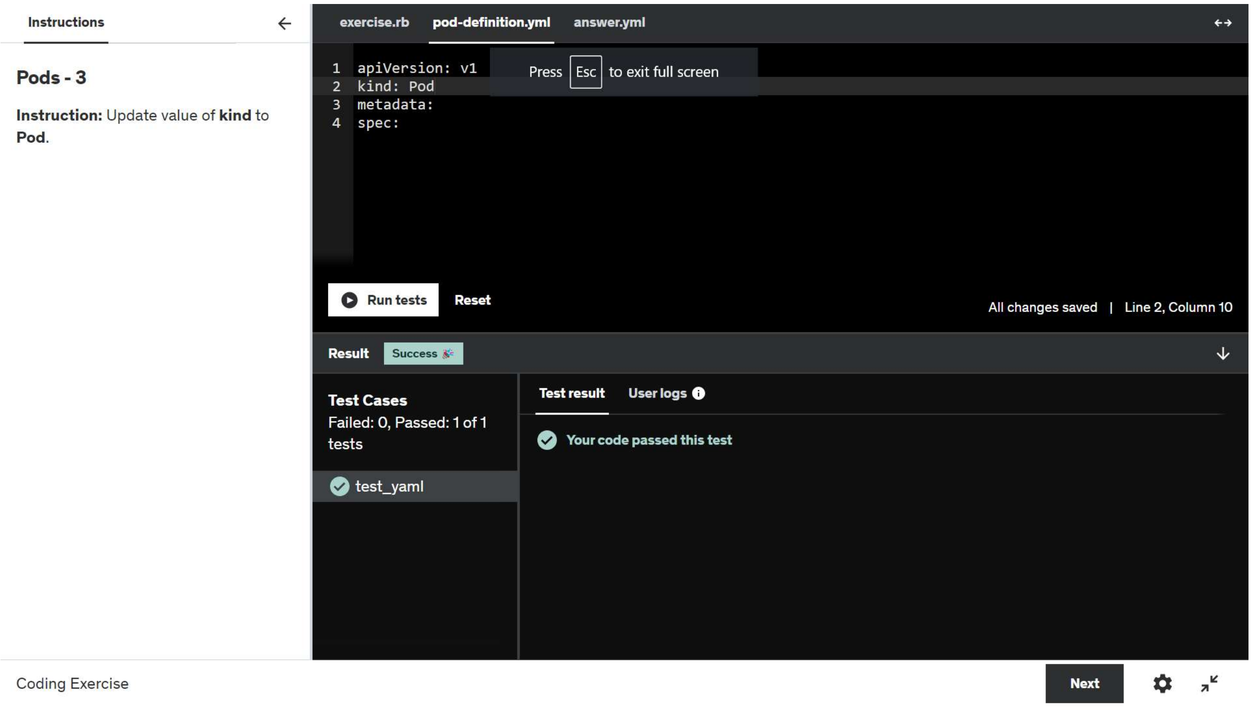
Instruction: Update value of **apiVersion** to **v1**. Remember to add a space between colon (:) and the value (v1)

```
apiVersion: v1
kind:
metadata:
spec:
```

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test



Pods - 3

Instruction: Update value of **kind** to **Pod**.

```yaml
apiVersion: v1
kind: Pod
metadata:
spec:
```

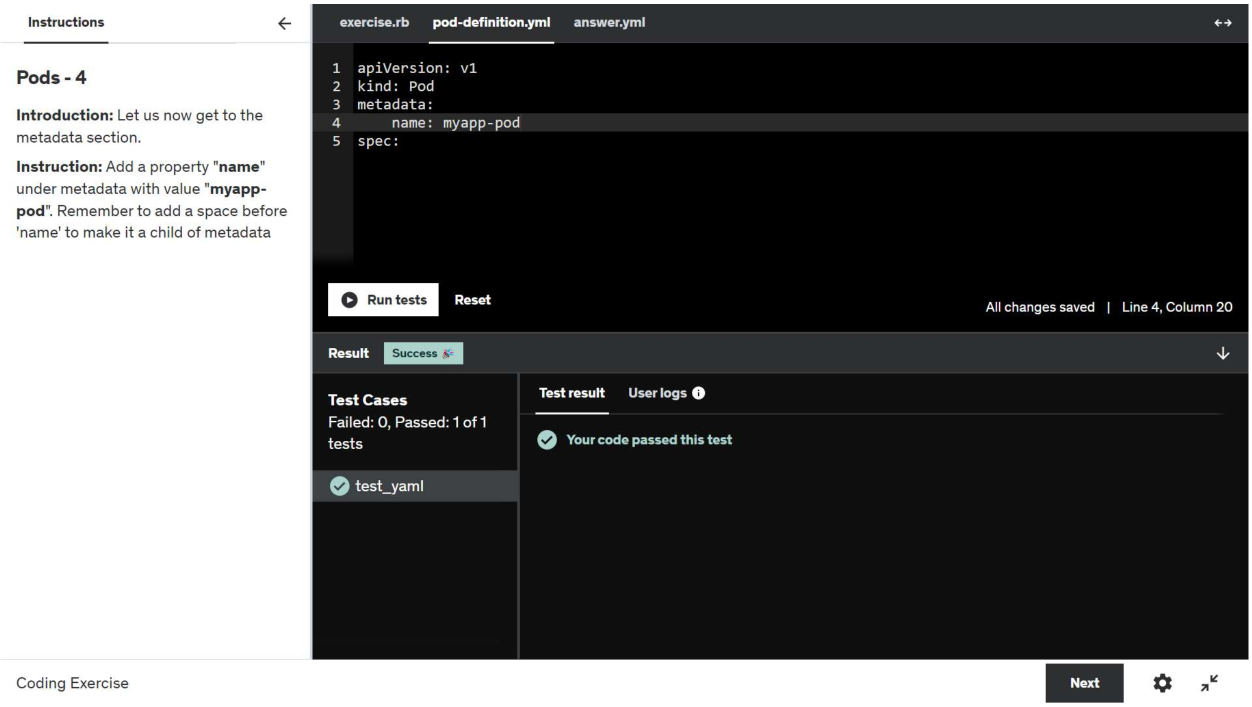
Test Cases

Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test

Coding Exercise



Pods - 4

Introduction: Let us now get to the metadata section.

Instruction: Add a property "**name**" under metadata with value "**myapp-pod**". Remember to add a space before 'name' to make it a child of metadata

```yaml
apiVersion: v1
kind: Pod
metadata:
    name: myapp-pod
spec:
```

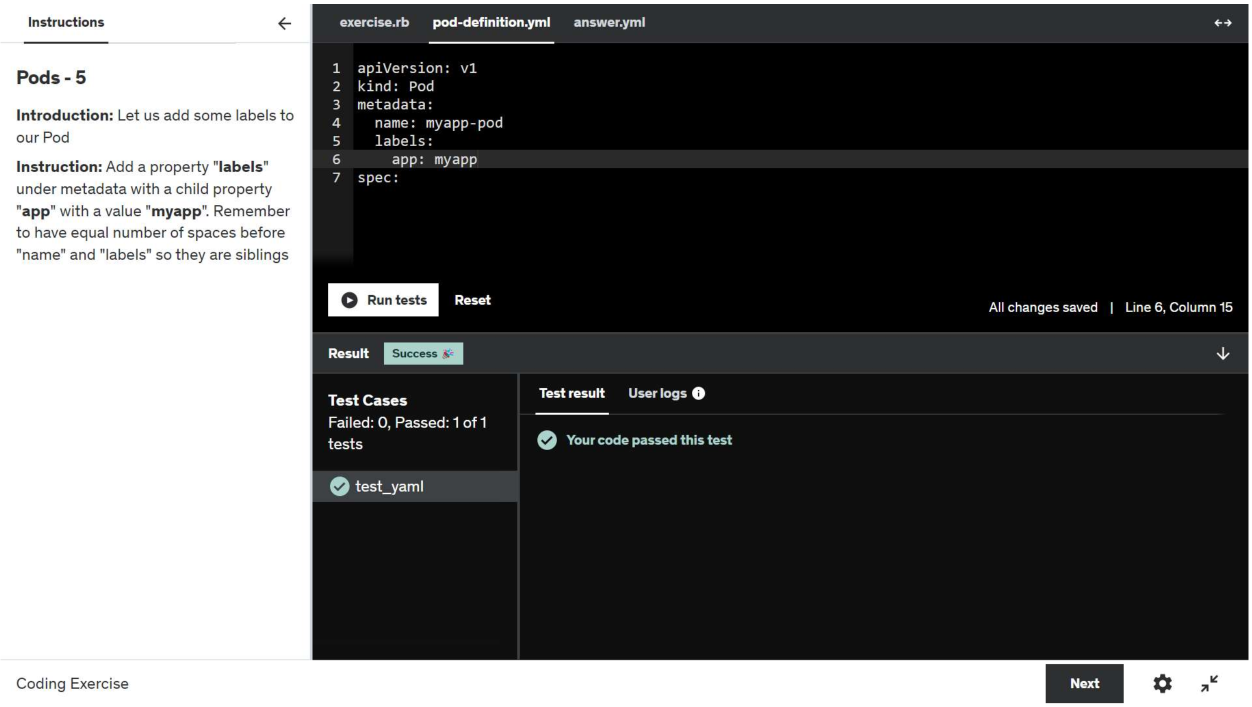
Test Cases

Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test

Coding Exercise



Pods - 5

Introduction: Let us add some labels to our Pod

Instruction: Add a property "**labels**" under metadata with a child property "**app**" with a value "**myapp**". Remember to have equal number of spaces before "name" and "labels" so they are siblings

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
```

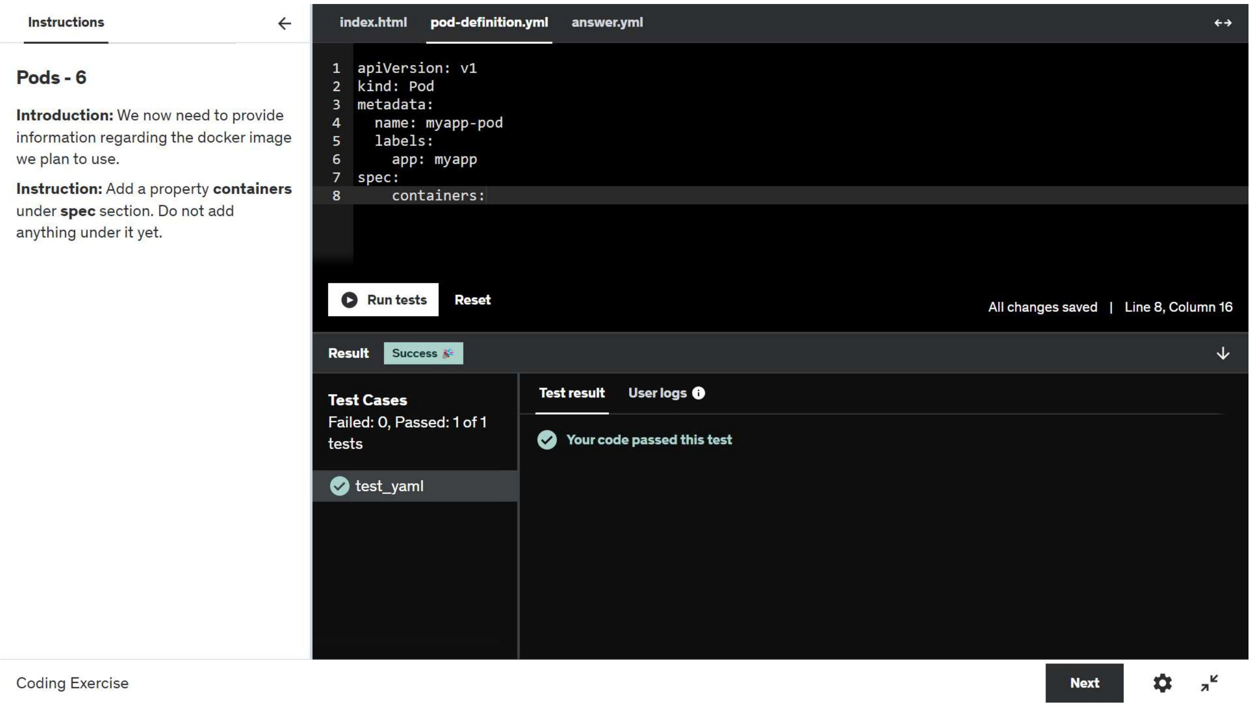
All changes saved | Line 6, Column 15

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test

Coding Exercise



Pods - 6

Introduction: We now need to provide information regarding the docker image we plan to use.

Instruction: Add a property **containers** under **spec** section. Do not add anything under it yet.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
    containers:
```

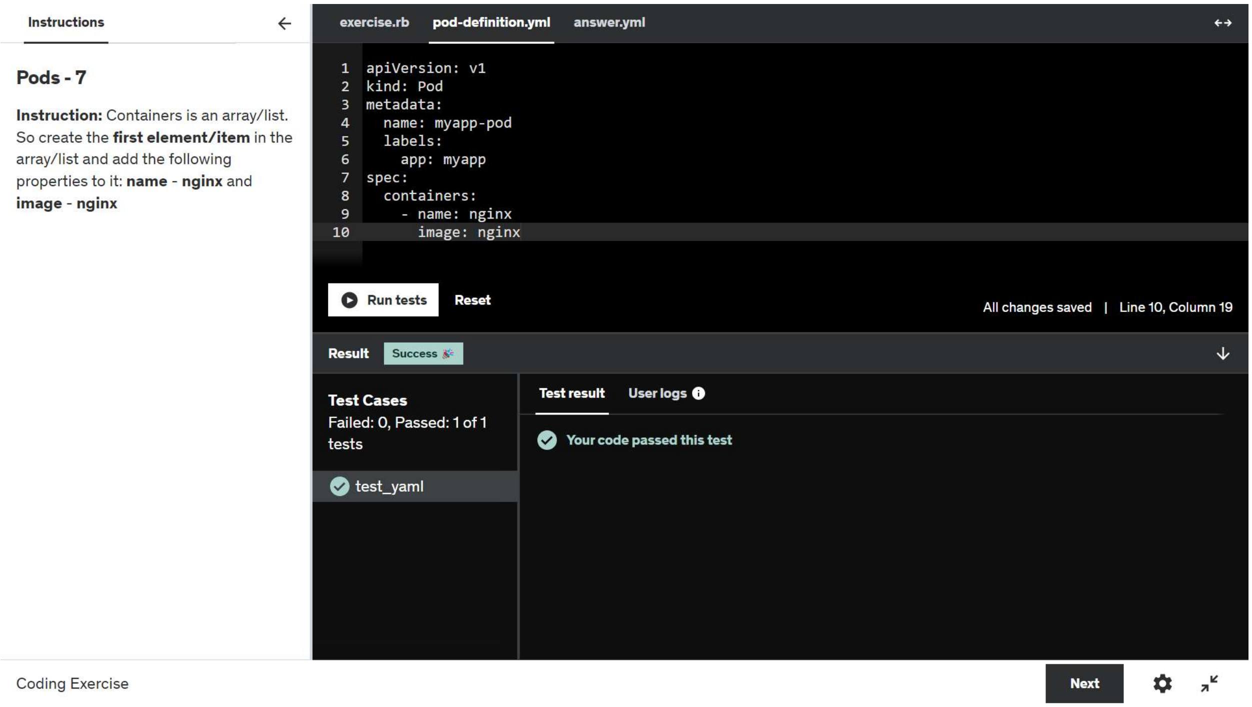
All changes saved | Line 8, Column 16

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test

Coding Exercise



Pods - 7

Instruction: Containers is an array/list. So create the **first element/item** in the array/list and add the following properties to it: **name** - **nginx** and **image** - **nginx**

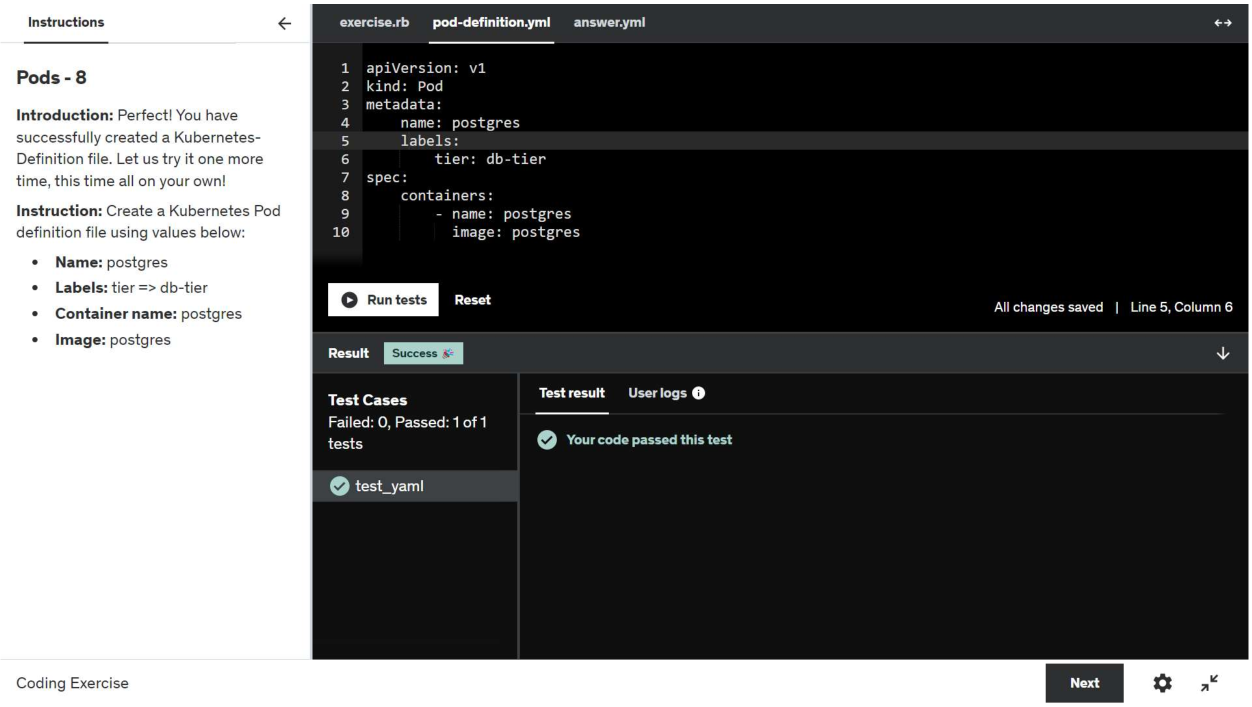
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
    - name: nginx
      image: nginx
```

Test Cases

Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test



Pods - 8

Introduction: Perfect! You have successfully created a Kubernetes-Definition file. Let us try it one more time, this time all on your own!

Instruction: Create a Kubernetes Pod definition file using values below:

- **Name:** postgres
- **Labels:** tier => db-tier
- **Container name:** postgres
- **Image:** postgres

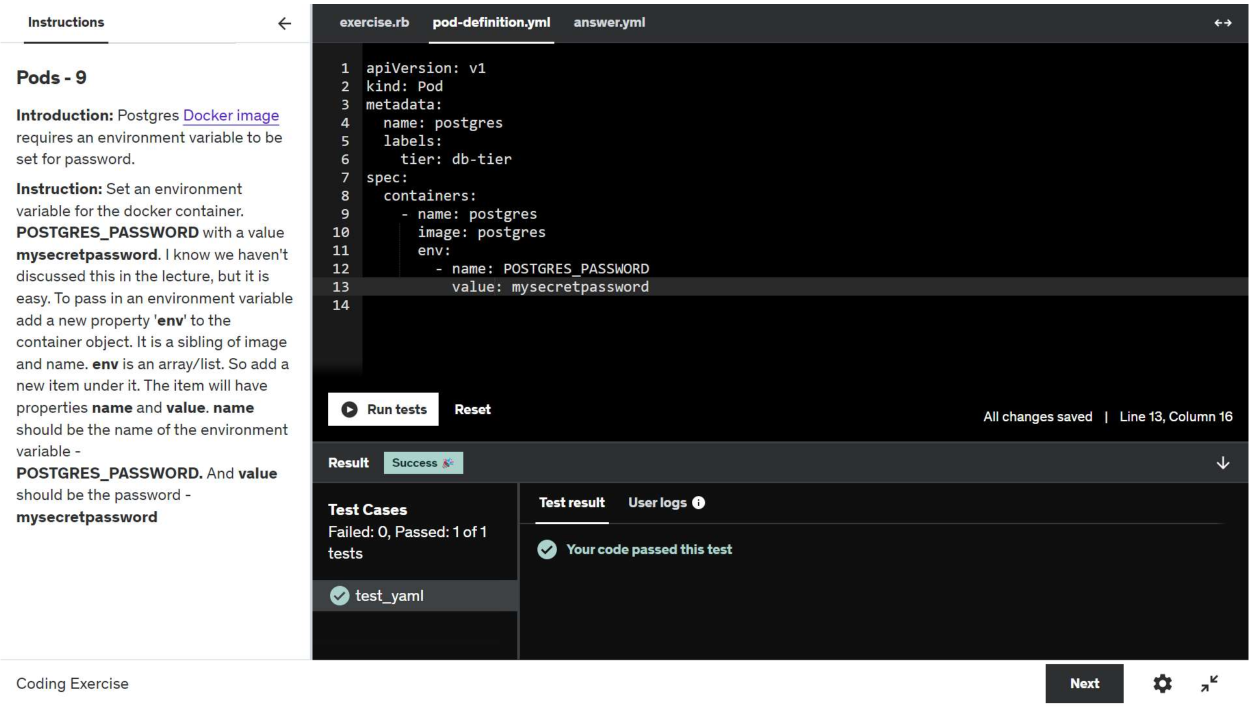
```yaml
apiVersion: v1
kind: Pod
metadata:
    name: postgres
    labels:
        tier: db-tier
spec:
    containers:
        - name: postgres
          image: postgres
```

Test Cases

Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test



Pods - 9

Introduction: Postgres Docker image requires an environment variable to be set for password.

Instruction: Set an environment variable for the docker container. **POSTGRES_PASSWORD** with a value **mysecretpassword**. I know we haven't discussed this in the lecture, but it is easy. To pass in an environment variable add a new property '**env**' to the container object. It is a sibling of image and name. **env** is an array/list. So add a new item under it. The item will have properties **name** and **value**. **name** should be the name of the environment variable - **POSTGRES_PASSWORD.** And **value** should be the password - **mysecretpassword**

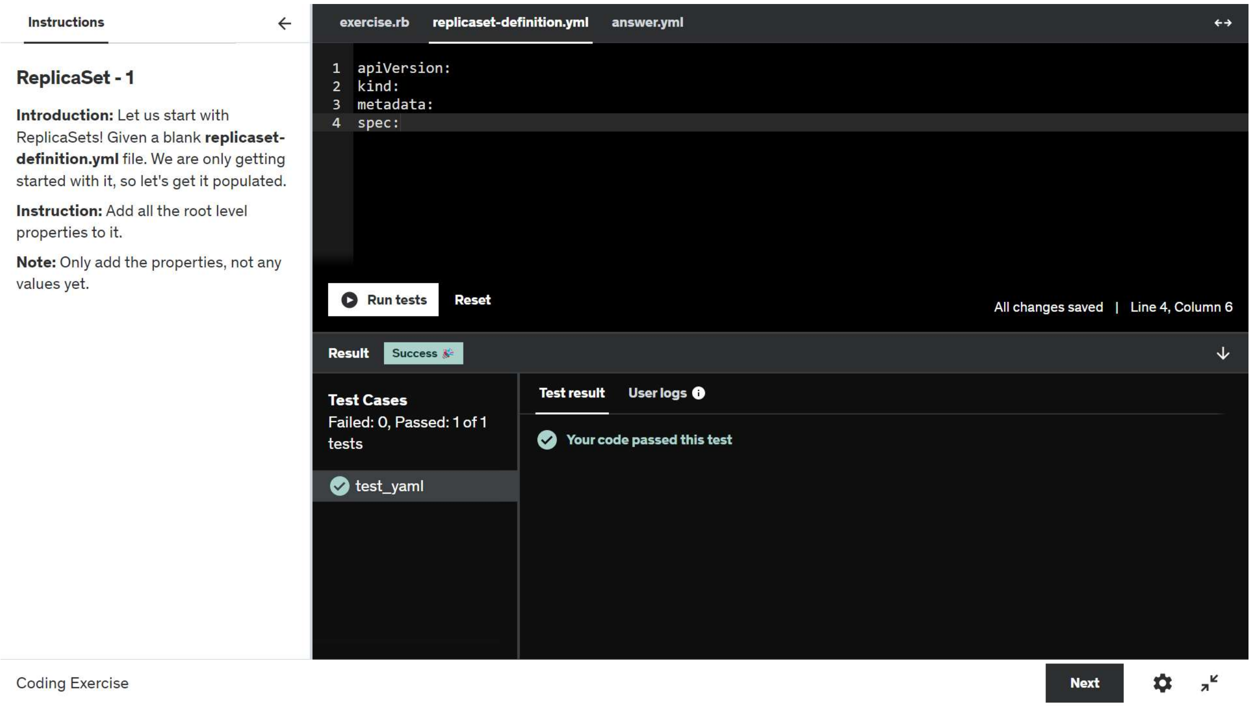
```
apiVersion: v1
kind: Pod
metadata:
  name: postgres
  labels:
    tier: db-tier
spec:
  containers:
    - name: postgres
      image: postgres
      env:
        - name: POSTGRES_PASSWORD
          value: mysecretpassword

```

All changes saved | Line 13, Column 16

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test



ReplicaSet - 1

Introduction: Let us start with ReplicaSets! Given a blank **replicaset-definition.yml** file. We are only getting started with it, so let's get it populated.

Instruction: Add all the root level properties to it.

Note: Only add the properties, not any values yet.

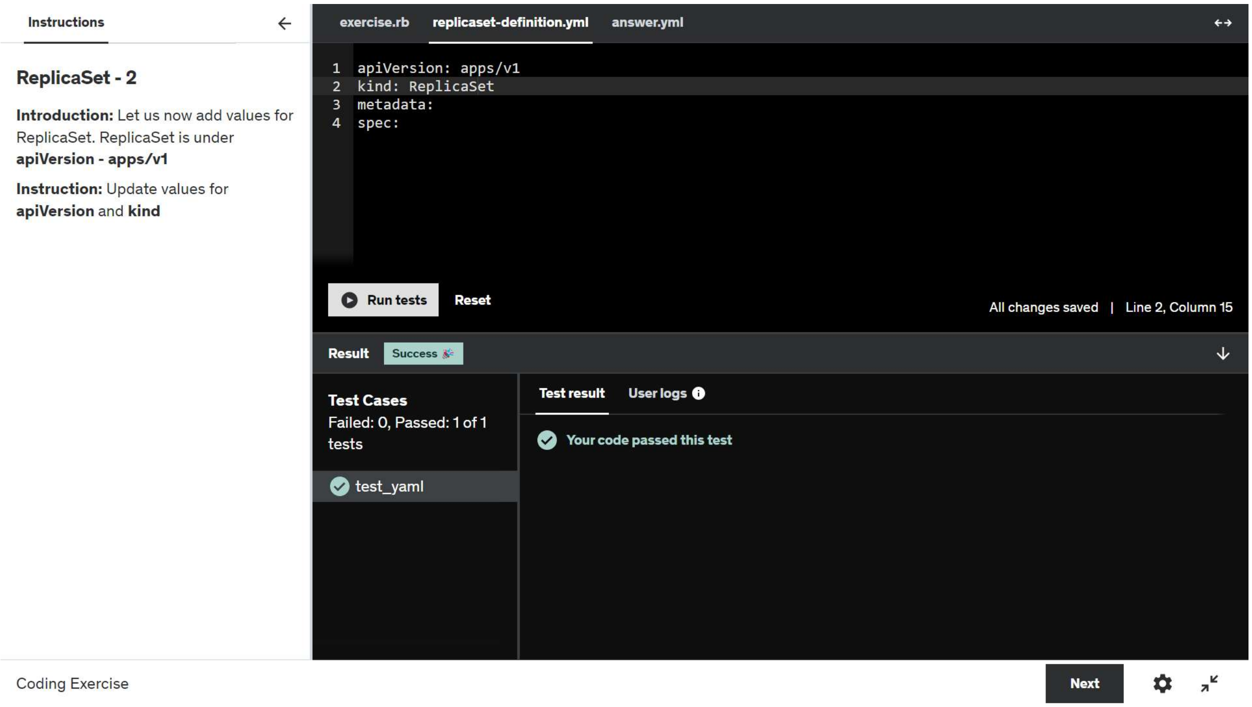
```
apiVersion:
kind:
metadata:
spec:
```

All changes saved | Line 4, Column 6

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test



ReplicaSet - 2

Introduction: Let us now add values for ReplicaSet. ReplicaSet is under **apiVersion - apps/v1**

Instruction: Update values for **apiVersion** and **kind**

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
spec:
```

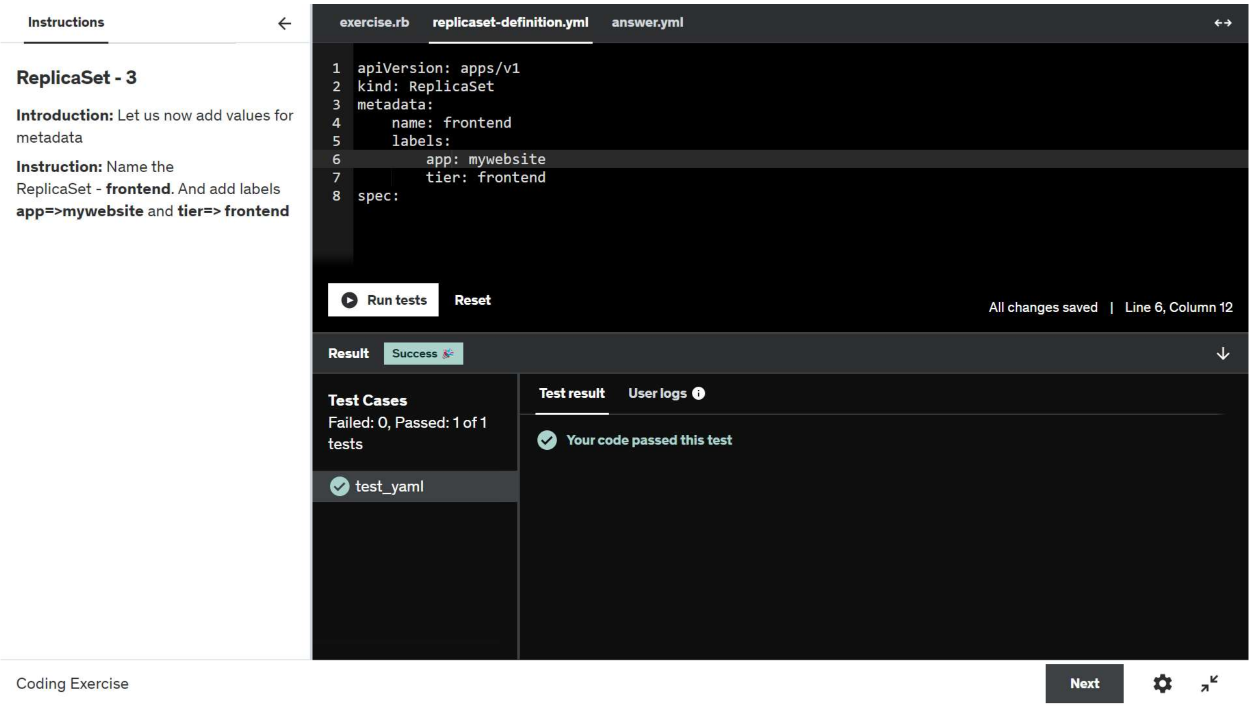
All changes saved | Line 2, Column 15

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test

Coding Exercise



ReplicaSet - 3

Introduction: Let us now add values for metadata

Instruction: Name the ReplicaSet - **frontend**. And add labels **app=>mywebsite** and **tier=> frontend**

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
    name: frontend
    labels:
        app: mywebsite
        tier: frontend
spec:
```

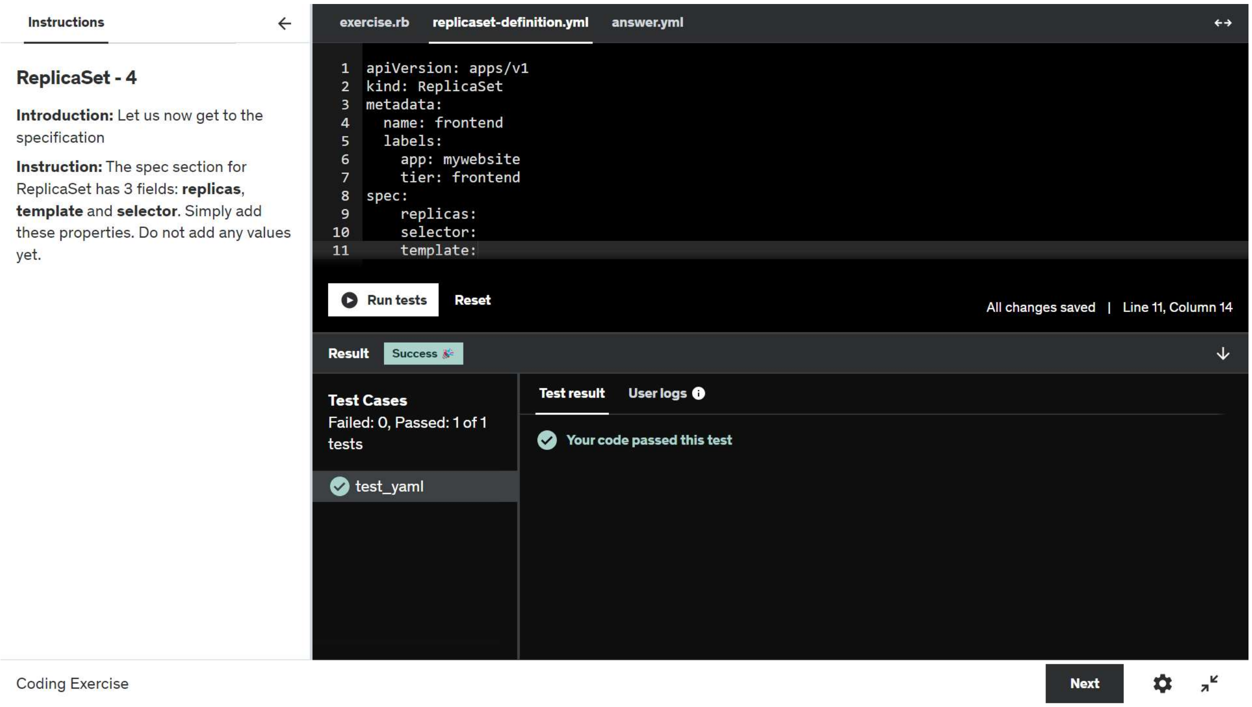
All changes saved | Line 6, Column 12

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test

Coding Exercise



ReplicaSet - 4

Introduction: Let us now get to the specification

Instruction: The spec section for ReplicaSet has 3 fields: **replicas**, **template** and **selector**. Simply add these properties. Do not add any values yet.

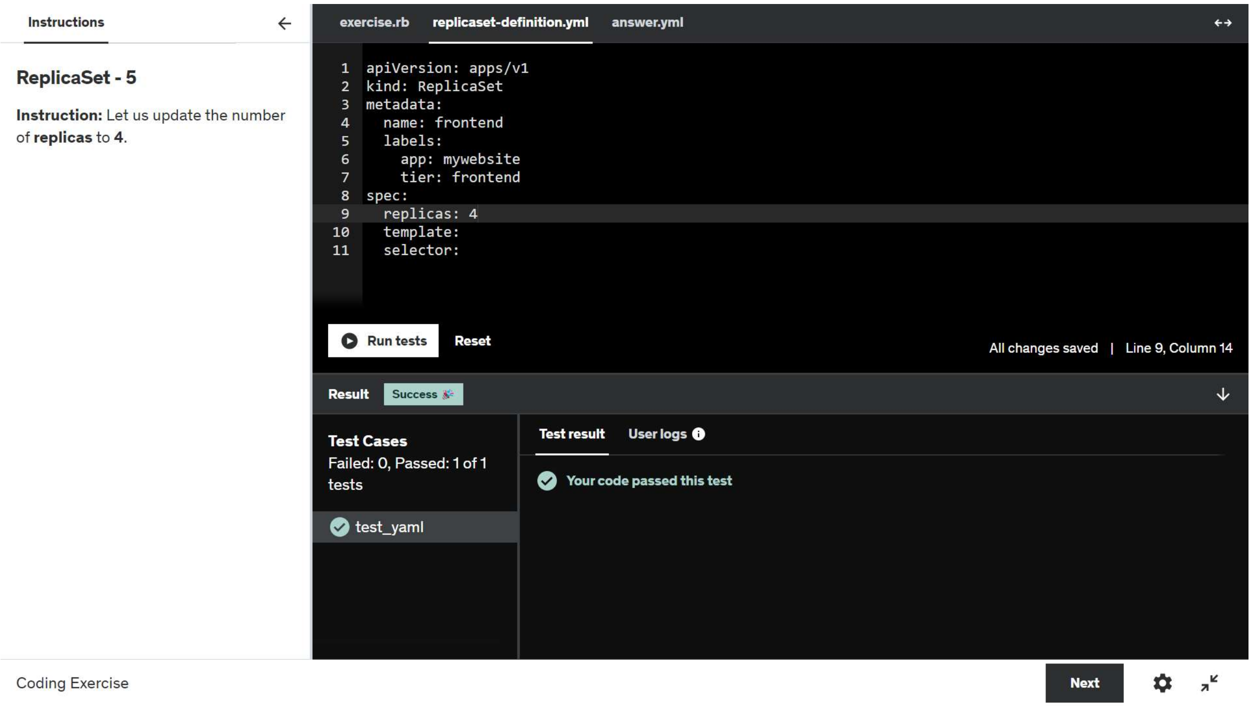
```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
    replicas:
    selector:
    template:
```

Test Cases

Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test



ReplicaSet - 5

Instruction: Let us update the number of **replicas** to **4**.

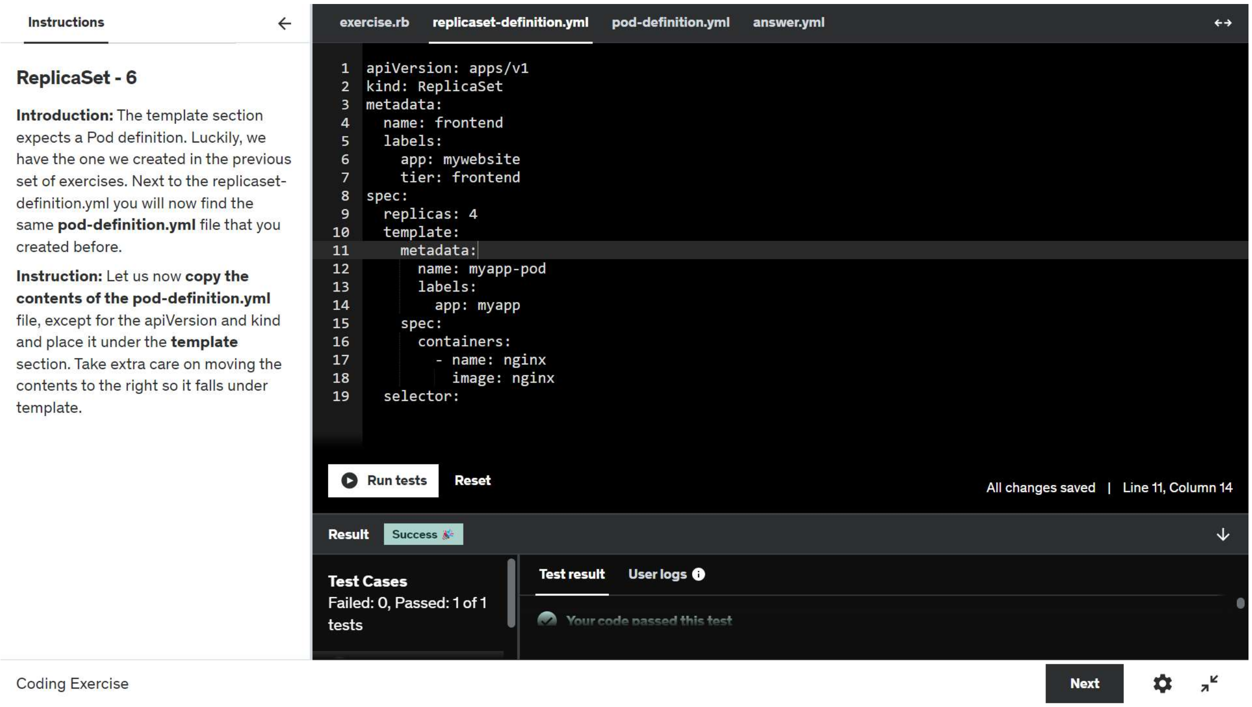
```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas: 4
  template:
  selector:
```

Test Cases

Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test



ReplicaSet - 6

Introduction: The template section expects a Pod definition. Luckily, we have the one we created in the previous set of exercises. Next to the replicaset-definition.yml you will now find the same **pod-definition.yml** file that you created before.

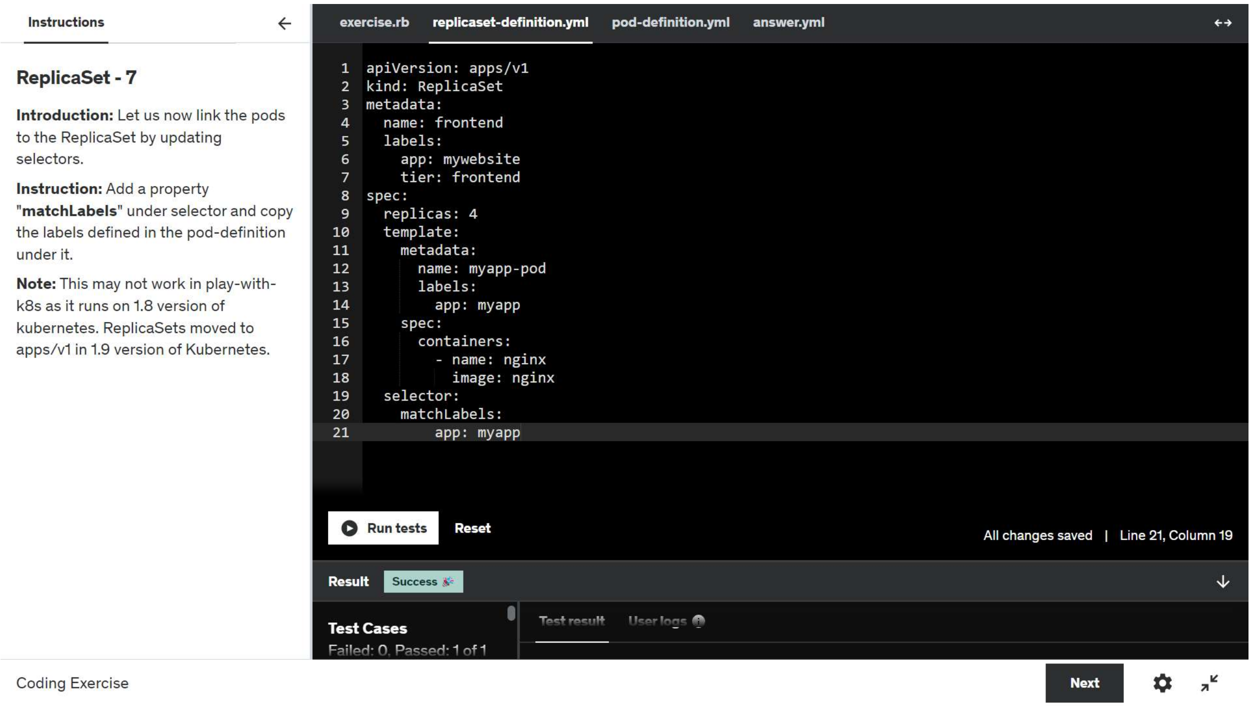
Instruction: Let us now **copy the contents of the pod-definition.yml** file, except for the apiVersion and kind and place it under the **template** section. Take extra care on moving the contents to the right so it falls under template.

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas: 4
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
    spec:
      containers:
        - name: nginx
          image: nginx
  selector:
```

All changes saved | Line 11, Column 14

Test Cases
Failed: 0, Passed: 1 of 1 tests

Your code passed this test



ReplicaSet - 7

Introduction: Let us now link the pods to the ReplicaSet by updating selectors.

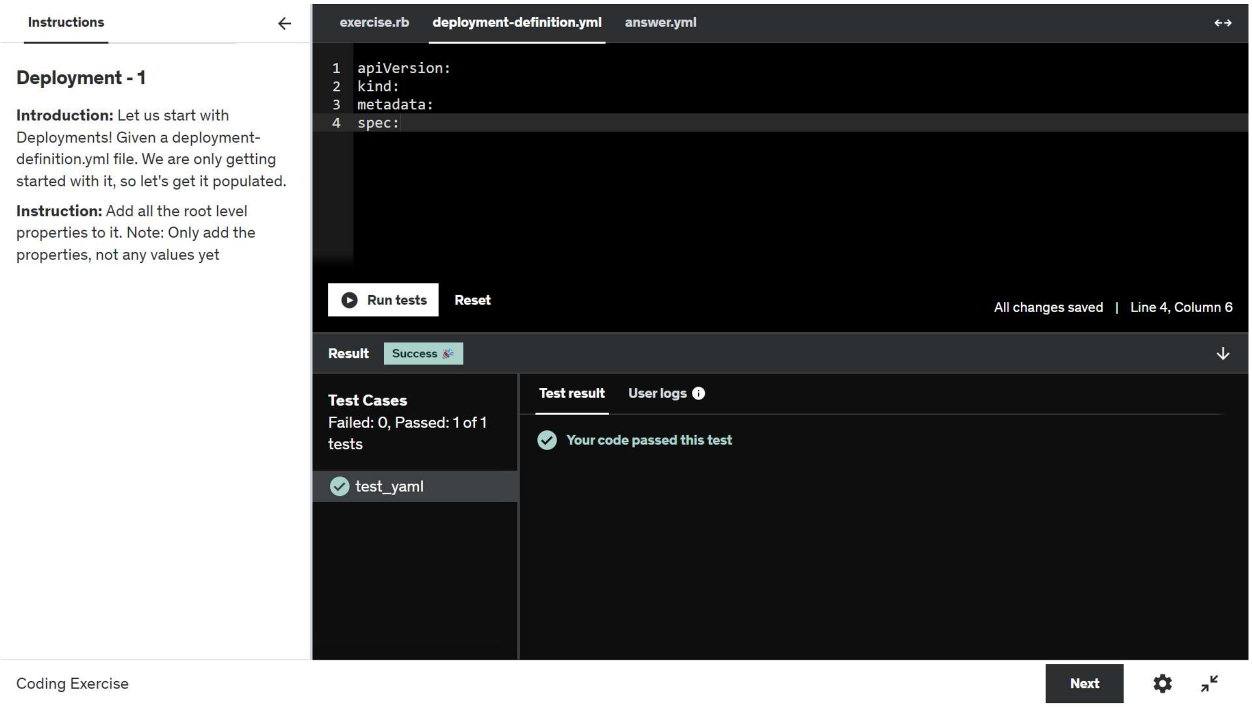
Instruction: Add a property "**matchLabels**" under selector and copy the labels defined in the pod-definition under it.

Note: This may not work in play-with-k8s as it runs on 1.8 version of kubernetes. ReplicaSets moved to apps/v1 in 1.9 version of Kubernetes.

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas: 4
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
    spec:
      containers:
        - name: nginx
          image: nginx
  selector:
    matchLabels:
        app: myapp
```

All changes saved | Line 21, Column 19

Test Cases
Failed: 0, Passed: 1 of 1



Deployment - 1

Introduction: Let us start with Deployments! Given a deployment-definition.yml file. We are only getting started with it, so let's get it populated.

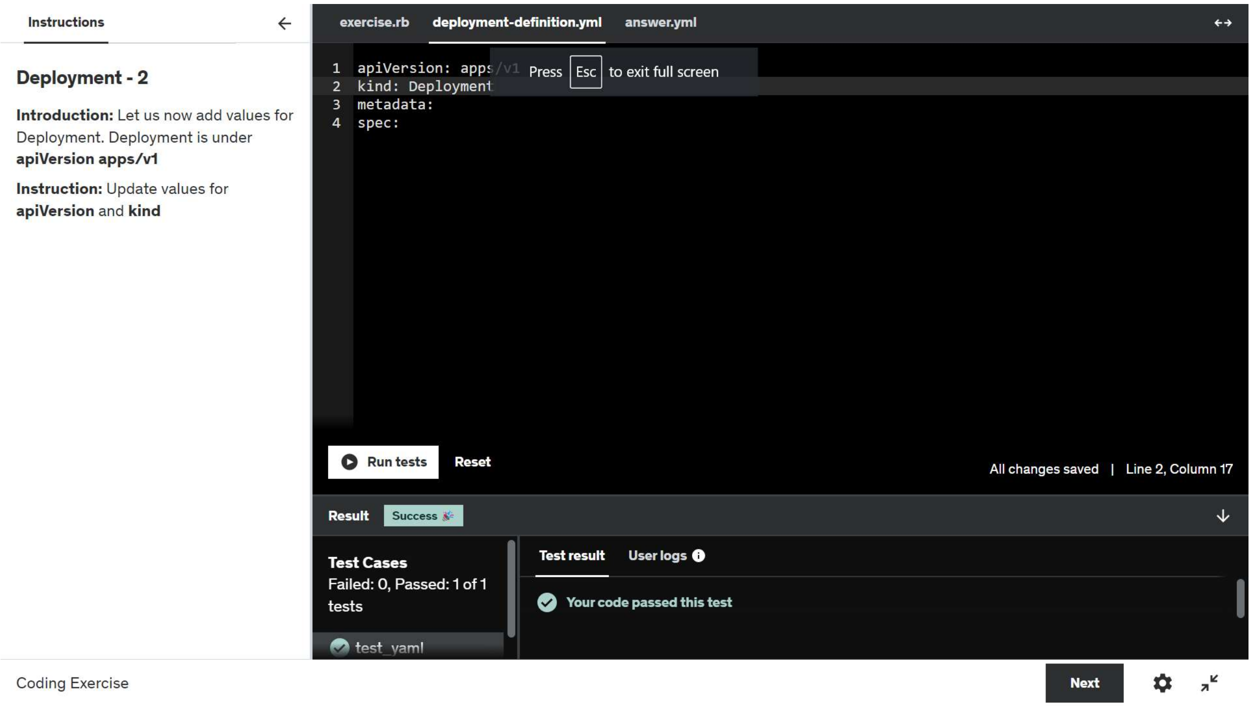
Instruction: Add all the root level properties to it. Note: Only add the properties, not any values yet

```
apiVersion:
kind:
metadata:
spec:
```

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test



Deployment - 2

Introduction: Let us now add values for Deployment. Deployment is under **apiVersion apps/v1**

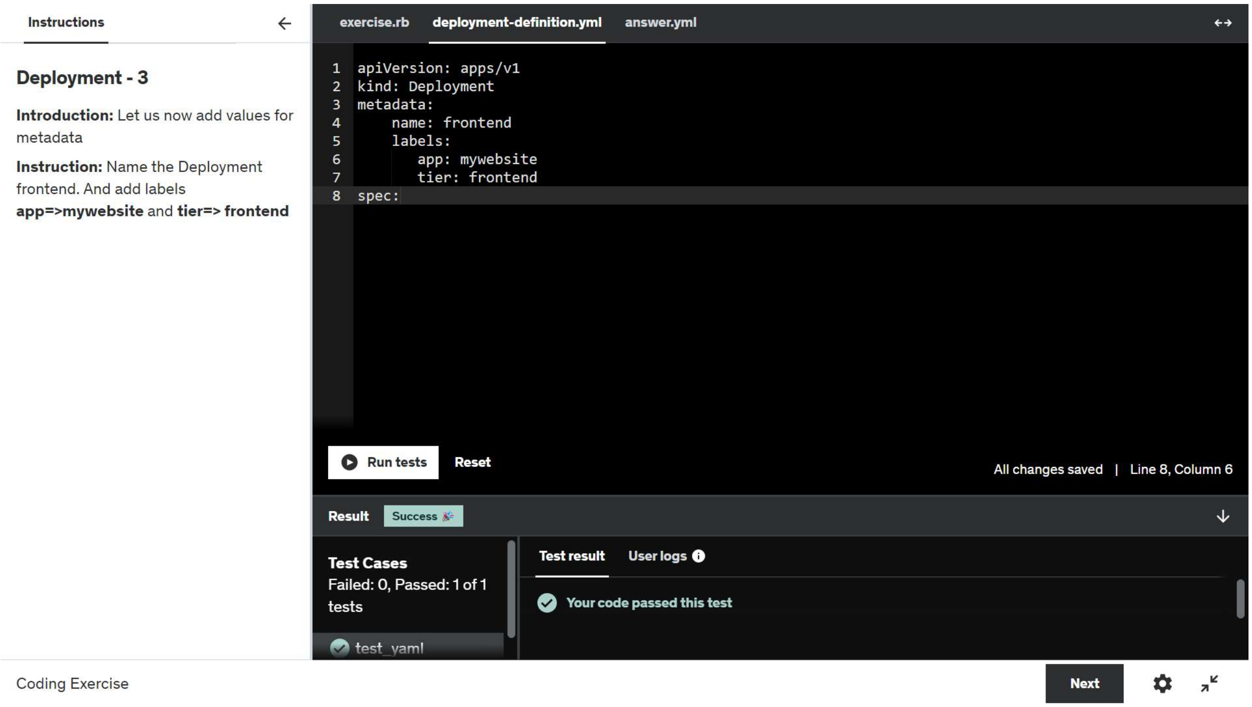
Instruction: Update values for **apiVersion** and **kind**

```
apiVersion: apps/v1
kind: Deployment
metadata:
spec:
```

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

✓ Your code passed this test



Deployment - 3

Introduction: Let us now add values for metadata

Instruction: Name the Deployment frontend. And add labels **app=>mywebsite** and **tier=> frontend**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
    name: frontend
    labels:
       app: mywebsite
       tier: frontend
spec:
```

All changes saved | Line 8, Column 6

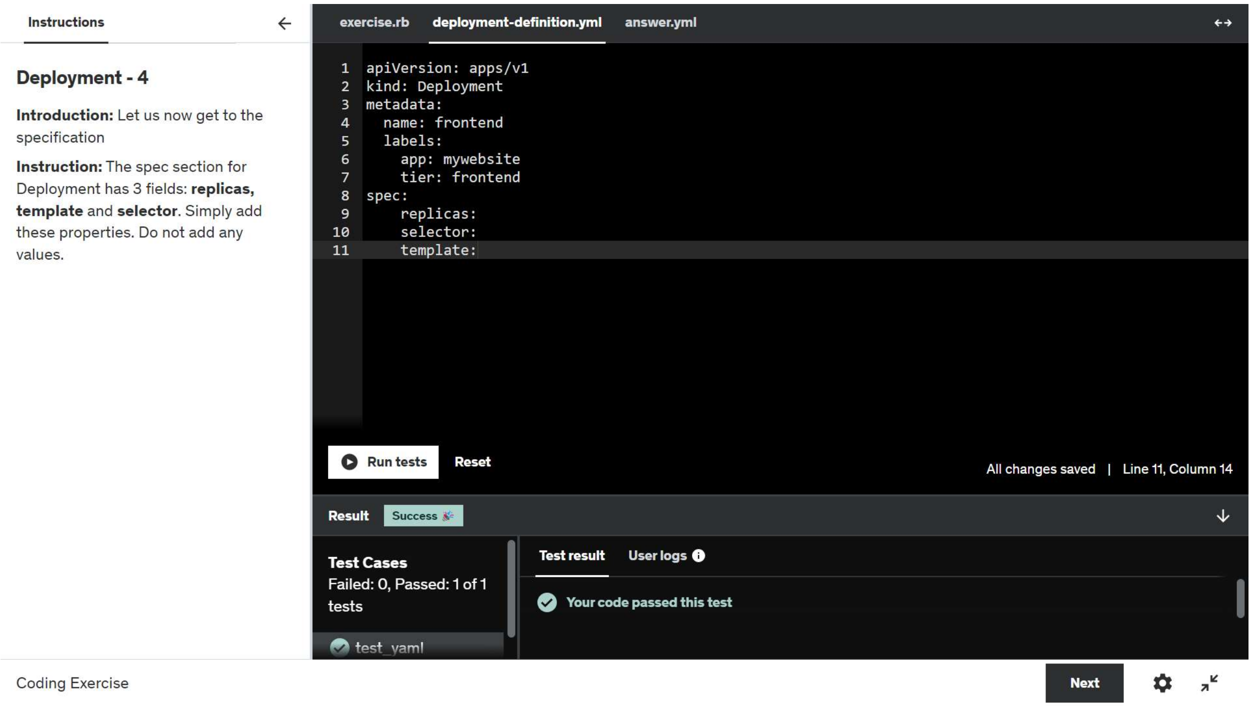
Success

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

Test result

✓ Your code passed this test



Deployment - 4

Introduction: Let us now get to the specification

Instruction: The spec section for Deployment has 3 fields: **replicas, template** and **selector**. Simply add these properties. Do not add any values.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
    replicas:
    selector:
    template:
```

All changes saved | Line 11, Column 14

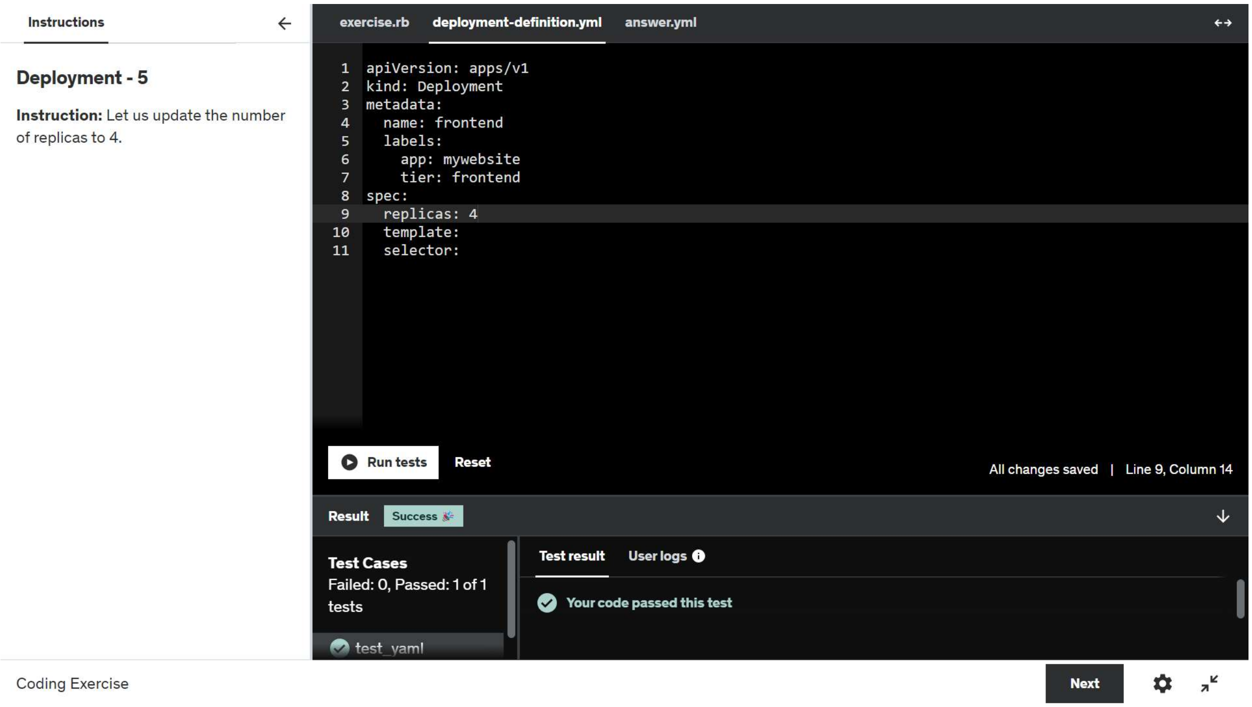
Success

Test Cases
Failed: 0, Passed: 1 of 1 tests

✓ test_yaml

Test result

✓ Your code passed this test



Deployment - 5

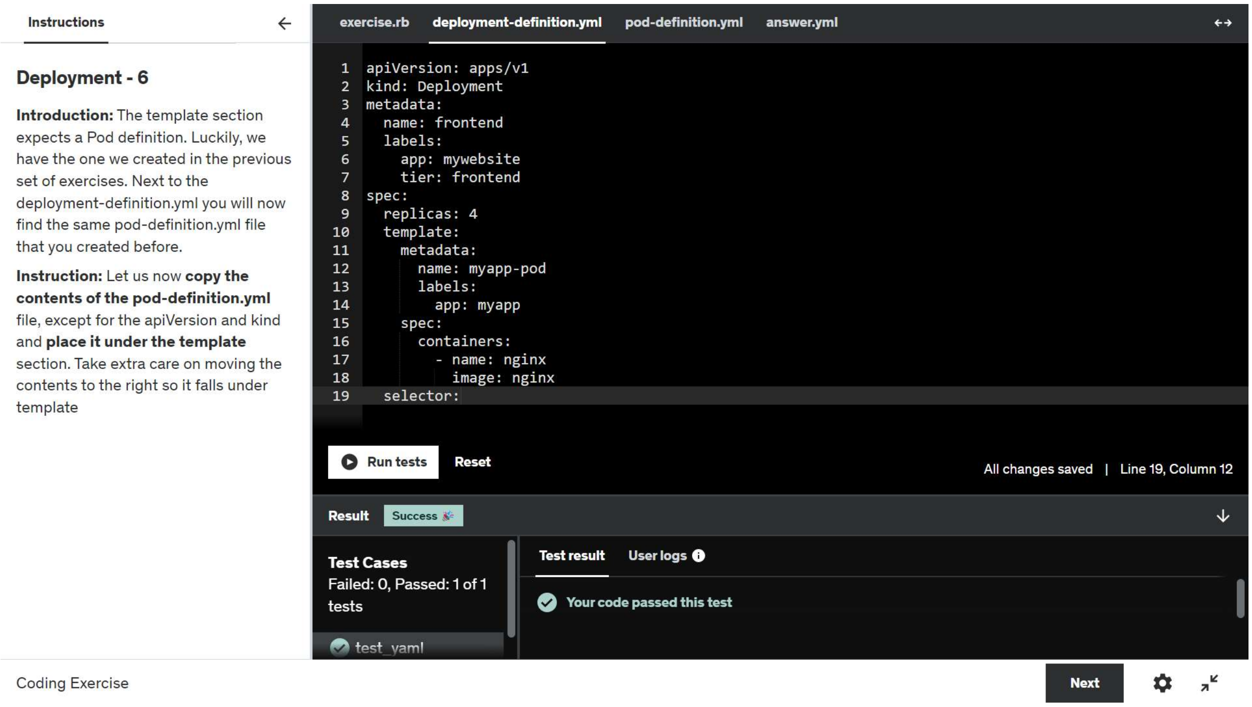
Instruction: Let us update the number of replicas to 4.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas: 4
  template:
  selector:
```

Test Cases
Failed: 0, Passed: 1 of 1 tests

test_yaml

Your code passed this test



Deployment - 6

Introduction: The template section expects a Pod definition. Luckily, we have the one we created in the previous set of exercises. Next to the deployment-definition.yml you will now find the same pod-definition.yml file that you created before.

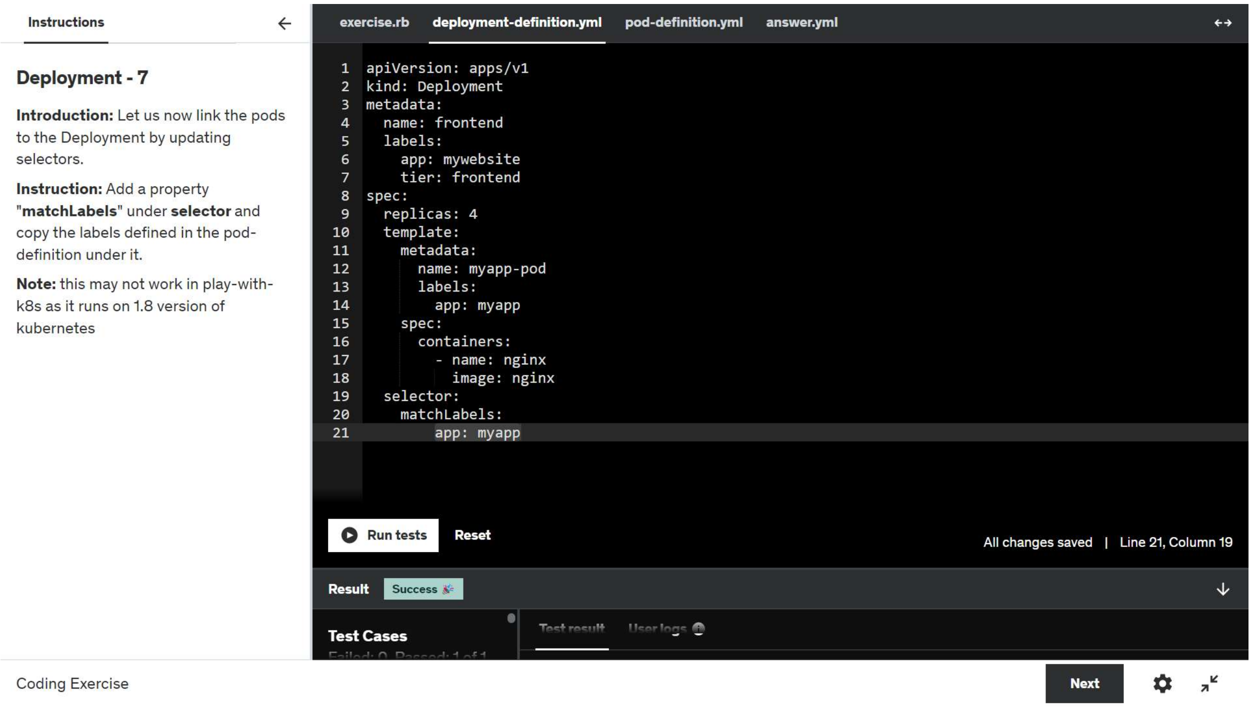
Instruction: Let us now **copy the contents of the pod-definition.yml** file, except for the apiVersion and kind and **place it under the template** section. Take extra care on moving the contents to the right so it falls under template

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas: 4
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
    spec:
      containers:
        - name: nginx
          image: nginx
  selector:
```

Test Cases
Failed: 0, Passed: 1 of 1 tests

test_yaml

Your code passed this test



Deployment - 7

Introduction: Let us now link the pods to the Deployment by updating selectors.

Instruction: Add a property "**matchLabels**" under **selector** and copy the labels defined in the pod-definition under it.

Note: this may not work in play-with-k8s as it runs on 1.8 version of kubernetes

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: mywebsite
    tier: frontend
spec:
  replicas: 4
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
    spec:
      containers:
        - name: nginx
          image: nginx
  selector:
    matchLabels:
        app: myapp
```