In [3]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

In [4]:
```
iris = load_iris()

print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

In [11]:
```
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42)

print(f"Training set size: {len(X_train)} samples")
print(f"Testing set size: {len(X_test)} samples")
```

```
Training set size: 120 samples
Testing set size: 30 samples
```

In [13]:
```
k = 3

knn_classifier = KNeighborsClassifier(n_neighbors=k)
knn_classifier.fit(X_train, y_train)

print(knn_classifier)
```

```
KNeighborsClassifier(n_neighbors=3)
```

In [14]:
```
y_pred = knn_classifier.predict(X_test)

print(y_pred)
```

```
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
```

In [15]:
```
from sklearn.metrics import classification_report

class_report = classification_report(y_test, y_pred, target_names=iris.target_names)
print("Classification Report:\n", class_report)
```

```
Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [18]:
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100: .2f}%")
```

```
Accuracy:  100.00%
```

In [19]:
```
k_values = [1, 3, 5, 7]

for k in k_values:
    knn_classifier = KNeighborsClassifier(n_neighbors=k)
    knn_classifier.fit(X_train, y_train)
    y_pred = knn_classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"k = {k}: Accuracy = {accuracy * 100: .2f}%")
```

```
k = 1: Accuracy =  100.00%
k = 3: Accuracy =  100.00%
k = 5: Accuracy =  100.00%
k = 7: Accuracy =  96.67%
```

In [27]:

```
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

from sklearn.model_selection import KFold
import numpy as np

classifiers = {
    'KNeighborsClassifier': KNeighborsClassifier(n_neighbors=3),
    'SVC': SVC(kernel='linear'),
    'GaussianNB': GaussianNB()
}

kf = KFold(n_splits=5, shuffle=True, random_state=42)

for name, classifier in classifiers.items():
    accuracies = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    classifier.fit(X_train, y_train)

    predictions = classifier.predict(X_test)

    accuracy = accuracy_score(y_test, predictions)
    accuracies.append(accuracy)

mean_accuracy = np.mean(accuracies)
std_accuracy = np.std(accuracies)

print(f"{name} Mean Accuracy: {mean_accuracy:.4f}")
print(f"{name} Standard Deviation: {std_accuracy:.4f}")
```

```
GaussianNB Mean Accuracy: 0.9600
GaussianNB Standard Deviation: 0.0249
```

In [30]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(y_test, y_pred)

class_labels = iris.target_names

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix,annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)

plt.xlabel('Predicted Labels')
plt.ylabel('Actual Labels')
plt.title('Confusion Matrix')
plt.show()
```