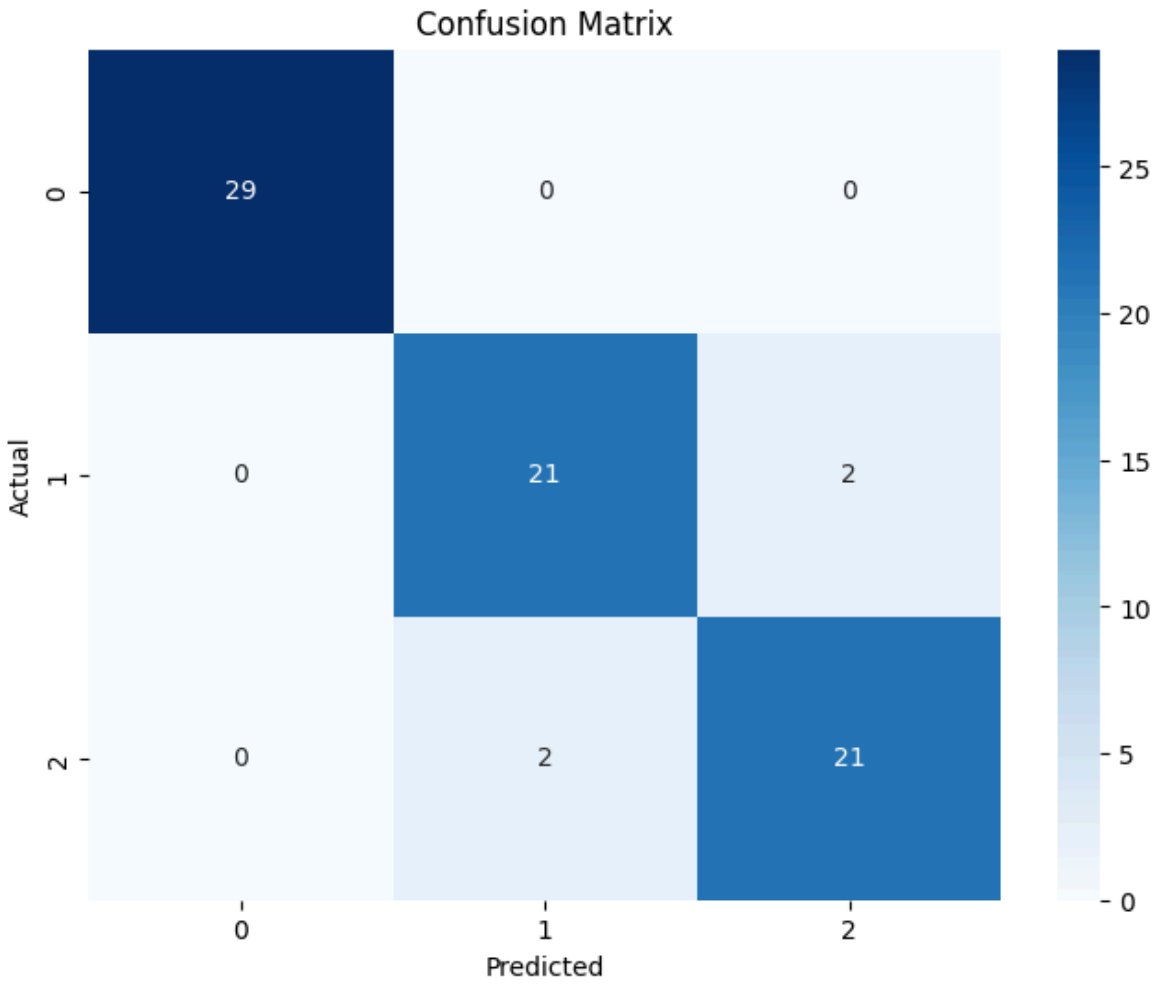
In [2]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confus
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import seaborn as sns

# Load the Iris dataset from Google Drive
iris_data = pd.read_csv('iris.csv') # Update with your file path

# Separate features (X) and target (y)
X = iris_data.drop('variety', axis=1)
y = iris_data['variety']

# Add noise to the features (to reduce accuracy)
noise_factor = 0.2  # Adjust this value to increase/decrease the noise le
X_noisy = X + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=X.

# Scale the noisy features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_noisy)

# Limit the training data size (to reduce model's learning capacity)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_siz

# Create a Gaussian Naive Bayes classifier
classifier = GaussianNB()

# Train the classifier
classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Test Accuracy: {accuracy:.2f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Compare Training and Testing Accuracy
y_train_pred = classifier.predict(X_train)
train_accuracy = accuracy_score(y_train, y_train_pred)
print(f"Training Accuracy: {train_accuracy:.2f}")

# Perform 5-fold Cross-Validation
cv_scores = cross_val_score(classifier, X_scaled, y, cv=5)
print(f"\nCross-Validation Accuracy: {cv_scores.mean():.2f} ± {cv_scores.

# Visualize the results (optional)
# Plotting the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blues', f
```

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

# Predict the species for a new data point (example)
# Get input from the user for sepal and petal measurements
sepal_length = float(input("Enter sepal length (cm): "))
sepal_width = float(input("Enter sepal width (cm): "))
petal_length = float(input("Enter petal length (cm): "))
petal_width = float(input("Enter petal width (cm): "))

# Create a list with the user's input
new_data = [[sepal_length, sepal_width, petal_length, petal_width]]
new_data_scaled = scaler.transform(new_data)  # Scale the new data
prediction = classifier.predict(new_data_scaled)
print(f"\nPrediction for new data: {prediction[0]}")
```

```
Test Accuracy: 0.95

Classification Report:
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        29
  Versicolor       0.91      0.91      0.91        23
   Virginica       0.91      0.91      0.91        23

    accuracy                           0.95        75
   macro avg       0.94      0.94      0.94        75
weighted avg       0.95      0.95      0.95        75


Confusion Matrix:
 [[29  0  0]
 [ 0 21  2]
 [ 0  2 21]]
Training Accuracy: 0.93

Cross-Validation Accuracy: 0.93 ± 0.04
```




```
Prediction for new data: Virginica
```

In [ ]: