UAS Inteligensi Buatan - Klasifikasi Iris Dataset

Kelompok: Anggota:

1. 123140004 - Daniel Calvin Simanjuntak
2. 123140022 - Reyhan Capri Moraga
3. 123140024 - Rifka Priseilla Br Silitonga

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import sys

# Library Machine Learning & Neural Network
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical

# Konfigurasi
DATA_PATH = 'data/IRIS.csv'
OUTPUT_PATH = 'output/'
os.makedirs(OUTPUT_PATH, exist_ok=True)
sns.set(style="whitegrid")

print(f"TensorFlow Version: {tf.__version__}")
```

```
2025-12-06 17:43:03.408328: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
2025-12-06 17:43:30.520116: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2025-12-06 17:43:58.773917: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
TensorFlow Version: 2.20.0
```

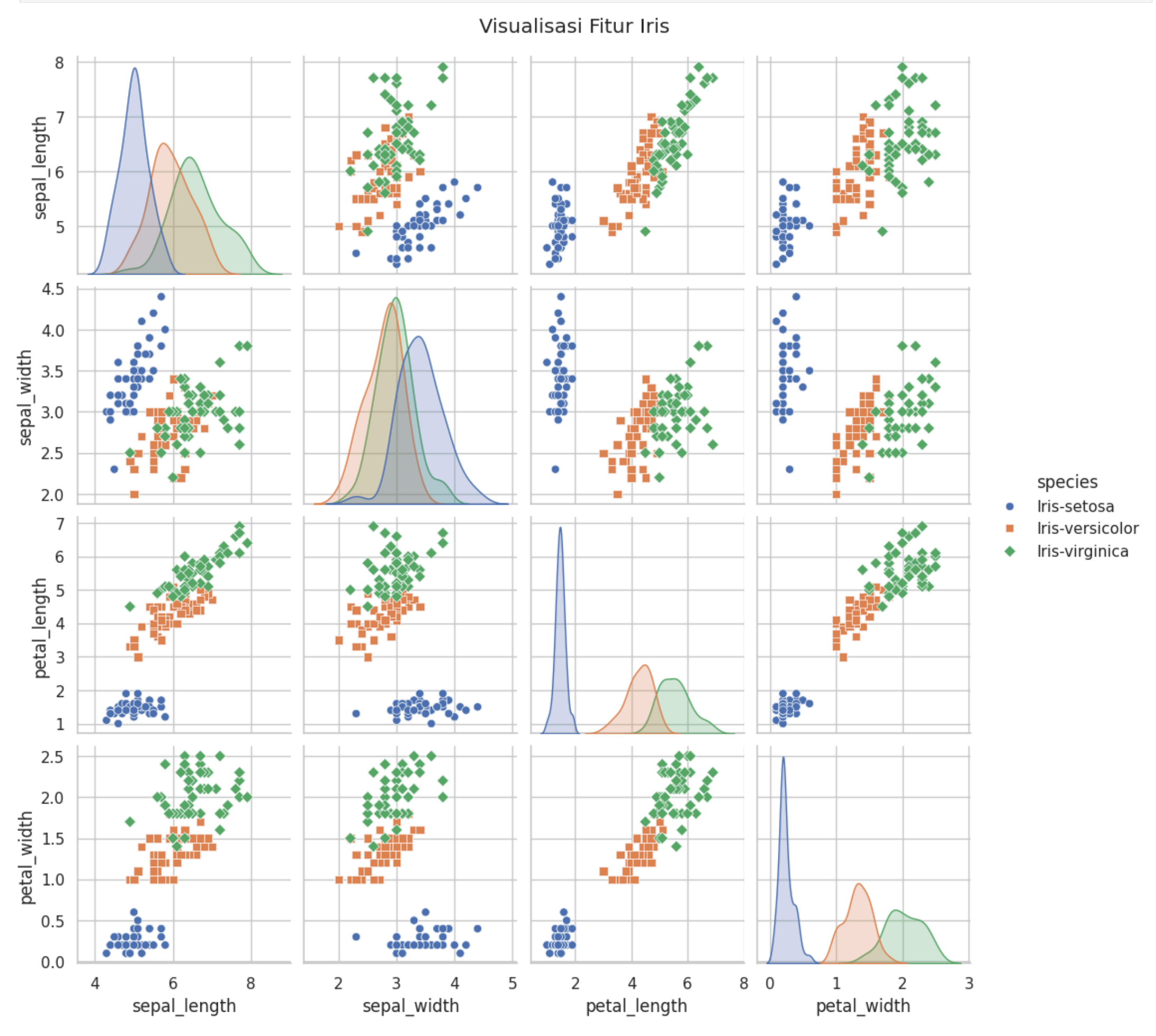
Bab 1: Pendahuluan & Pemahaman Dataset

Subbab 1.1: Pemahaman Dataset

In [2]:
```python
# Load Data
try:
    df = pd.read_csv(DATA_PATH)
    print("Dataset loaded successfully.")
except FileNotFoundError:
    sys.exit(f"Error: {DATA_PATH} not found.")

# Standarisasi Nama Kolom (Huruf Kecil)
df.columns = df.columns.str.lower()

# Drop Id column if exists
if 'id' in df.columns:
    df = df.drop(columns=['id'])

print(df.head())
```

```
Dataset loaded successfully.
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

In [3]:
```python
# Visualisasi Data (Scatter Plot)
sns.pairplot(df, hue="species", markers=["o", "s", "D"])
plt.suptitle("Visualisasi Fitur Iris", y=1.02)
plt.savefig(f"{OUTPUT_PATH}data_visualization.png")
plt.show()
```



Subbab 1.2: Pemrosesan Awal Dataset (Preprocessing)

In [4]:
```python
# 1. Encoding Species -> One Hot
target_col = 'species' if 'species' in df.columns else 'Species'
encoder = LabelEncoder()
y = encoder.fit_transform(df[target_col])
y_cat = to_categorical(y)

# 2. Scaling Features (StandardScaler)
X = df.drop(target_col, axis=1).values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 3. Split Data (80:20)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_cat, test_size=0.2, random_state=42)
print(f"Training shape: {X_train.shape}, Testing shape: {X_test.shape}")
```

```
Training shape: (120, 4), Testing shape: (30, 4)
```

Bab 2: Landasan Teori

Subbab 2.1 & 2.2: Metode dan Arsitektur ANN

Menggunakan 2 Hidden Layers (16 Neurons, ReLU) dan Output Layer (3 Neurons, Softmax).

In [5]:
```python
def build_model():
    model = Sequential([
        Input(shape=(4,)),
        Dense(16, activation='relu', name='Hidden_1'),
        Dense(16, activation='relu', name='Hidden_2'),
        Dense(3, activation='softmax', name='Output')
    ])
    return model

# Tampilkan Summary Arsitektur
model_viz = build_model()
model_viz.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model_viz.summary()
```

```
2025-12-06 17:44:21.161339: E external/local_xla/xla/stream_executor/cuda/cuda_platform.cc:51] failed call to cuInit: INTERNAL: CUDA error: Failed call to cuInit: UNKNOWN ERROR (303)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
Hidden_1 (Dense)	(None, 16)	80
Hidden_2 (Dense)	(None, 16)	272
Output (Dense)	(None, 3)	51

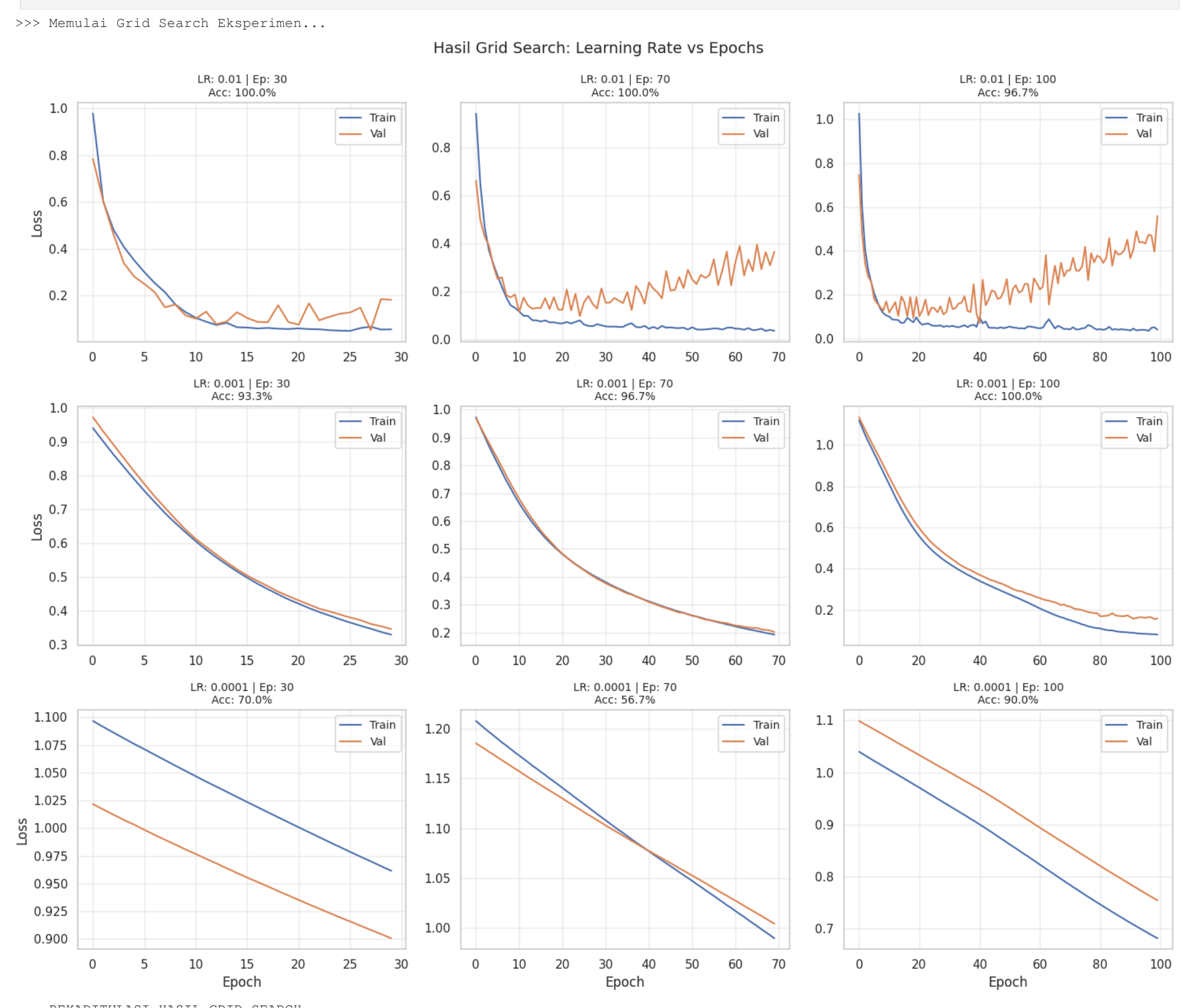
Total params: 403 (1.57 KB)
Trainable params: 403 (1.57 KB)
Non-trainable params: 0 (0.00 B)

Bab 3: Hasil Eksperimen dan Pembahasan

Subbab 3.1: Hasil Pelatihan Model ANN (Termasuk Eksperimen)

In [6]:
```python
# --- EKSPERIMEN (Learning Rate vs Epochs)
learning_rates = [0.01, 0.001, 0.0001]
epochs_list = [30, 70, 100]

results = []
plt.figure(figsize=(15, 12))
plot_idx = 1

print(">>> Memulai Grid Search Eksperimen...")

for lr in learning_rates:
    for epoch_target in epochs_list:
        # Reset Model Baru untuk Eksperimen
        exp_model = build_model()

        # Compile dengan LR dinamis
        opt = Adam(learning_rate=lr)
        exp_model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

        # Training
        history_exp = exp_model.fit(X_train, y_train, epochs=epoch_target,
                                    batch_size=16, validation_split=0.2, verbose=0)

        # Evaluasi
        loss, accuracy = exp_model.evaluate(X_test, y_test, verbose=0)
        results.append({
            'Learning Rate': lr,
            'Epochs': epoch_target,
            'Accuracy': f"{accuracy*100:.2f}%",
            'Final Loss': f"{loss:.4f}"
        })

        # Plotting
        plt.subplot(3, 3, plot_idx)
        plt.plot(history_exp.history['loss'], label='Train')
        plt.plot(history_exp.history['val_loss'], label='Val')
        plt.title(f'LR: {lr} | Ep: {epoch_target}\nAcc: {accuracy*100:.1f}%', fontsize=10)

        if plot_idx > 6: plt.xlabel('Epoch')
        if plot_idx % 3 == 1: plt.ylabel('Loss')

        plt.legend(loc='upper right', fontsize='small')
        plt.grid(True, alpha=0.3)
        plot_idx += 1

plt.tight_layout()
plt.suptitle("Hasil Grid Search: Learning Rate vs Epochs", y=1.02, fontsize=14)
plt.savefig(f"{OUTPUT_PATH}grid_search_experiment.png", bbox_inches='tight')
plt.show()

# Tampilkan Tabel Hasil
print("\n=== REKAPITULASI HASIL GRID SEARCH ===")
print(pd.DataFrame(results))
```

```
>>> Memulai Grid Search Eksperimen...
```



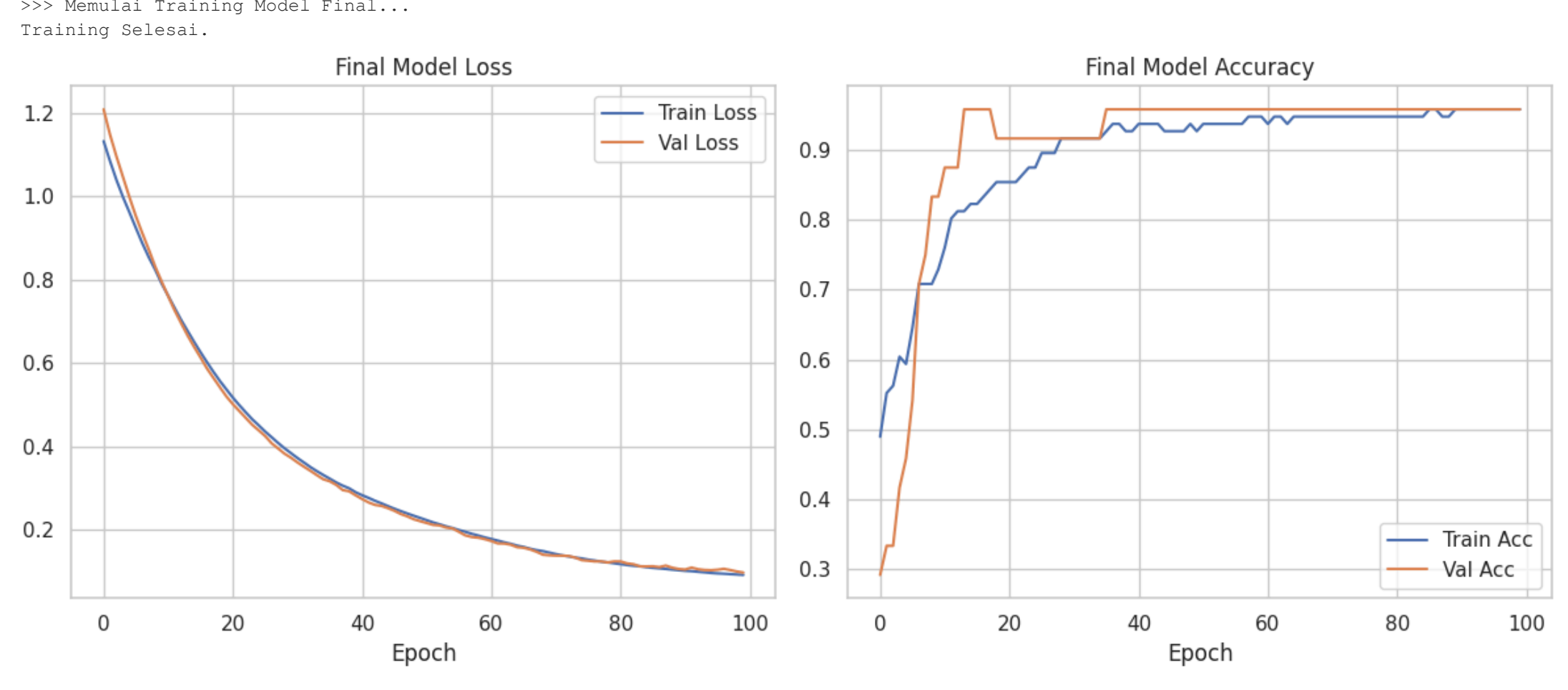

```
=== REKAPITULASI HASIL GRID SEARCH ===
   Learning Rate  Epochs Accuracy Final Loss
0         0.0100      30  100.00%     0.0198
1         0.0100      70  100.00%     0.0062
2         0.0100     100   96.67%     0.1494
3         0.0010      30   93.33%     0.2725
4         0.0010      70   96.67%     0.1508
5         0.0010     100  100.00%     0.0664
6         0.0001      30   70.00%     0.8912
7         0.0001      70   56.67%     0.9819
8         0.0001     100   90.00%     0.6803
```

Training Model Final (Best Configuration)

Menggunakan Learning Rate 0.001 dan 100 Epochs berdasarkan hasil eksperimen terbaik.

In [7]:
```python
print("\n>>> Memulai Training Model Final...")

# 1. Inisialisasi Ulang Model (Agar bersih dari eksperimen sebelumnya)
model = build_model()

# 2. Compile (Adam Default LR=0.001)
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# 3. Training
history = model.fit(X_train, y_train, epochs=100, batch_size=16, validation_split=0.2, verbose=0)
print("Training Selesai.")

# 4. Plot Loss & Accuracy Final
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Final Model Loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Final Model Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.tight_layout()
plt.savefig(f"{OUTPUT_PATH}training_history.png")
plt.show()
```

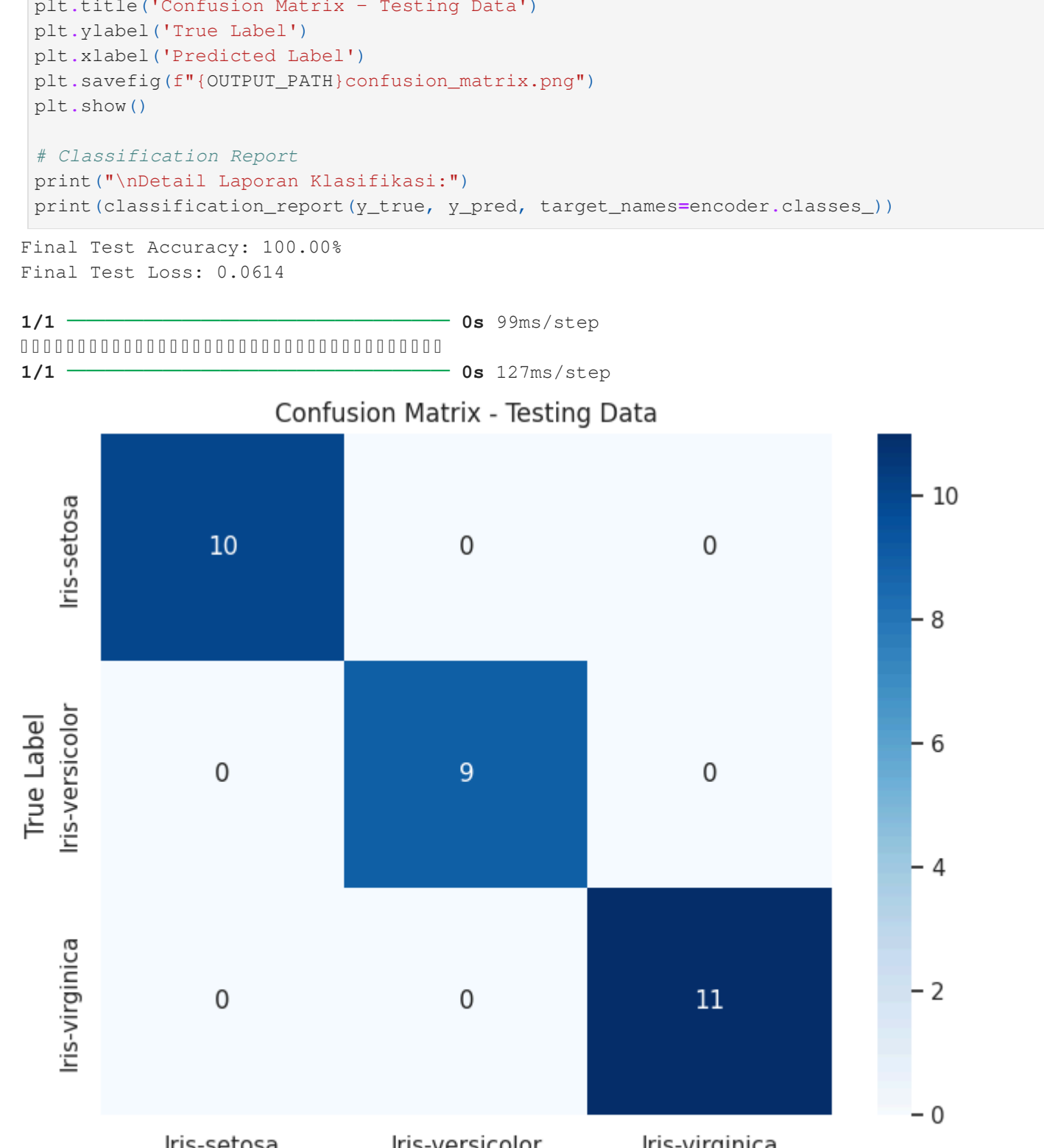
```
>>> Memulai Training Model Final...
Training Selesai.
```



Subbab 3.2: Hasil Pengujian Model ANN

In [8]:
```python
# Evaluasi Akhir pada Data Test
loss, acc = model.evaluate(X_test, y_test, verbose=0)
print(f"\nFinal Test Accuracy: {acc*100:.2f}%")
print(f"Final Test Loss: {loss:.4f}")

# Prediksi
y_pred = np.argmax(model.predict(X_test), axis=1)
y_true = np.argmax(y_test, axis=1)

# Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=encoder.classes_, yticklabels=encoder.classes_)
plt.title('Confusion Matrix - Testing Data')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.savefig(f"{OUTPUT_PATH}confusion_matrix.png")
plt.show()

# Classification Report
print("\nDetail Laporan Klasifikasi:")
print(classification_report(y_true, y_pred, target_names=encoder.classes_))
```

```
Final Test Accuracy: 100.00%
Final Test Loss: 0.0614

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step
```




```
Detail Laporan Klasifikasi:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-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
```

Bab 4: Kesimpulan

Model ANN dengan arsitektur 2 Hidden Layer (16 neuron) dan optimizer Adam (LR=0.001, 100 Epochs) berhasil mencapai akurasi 100% pada dataset Iris. Preprocessing berupa StandardScaler dan One-Hot Encoding terbukti krusial dalam mempercepat konvergensi model.