Q1: Import / Load

In [15]:
```
# Step 1: Import libraries and set random seeds
import numpy as np
import tensorflow as tf
from tensorflow.keras import Sequential, layers
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

# Set random seed for reproducibility
tf.random.set_seed(42)
np.random.seed(42)

(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

label_names = ["T-shirt/top","Trouser","Pullover","Dress","Coat",
               "Sandal","Shirt","Sneaker","Bag","Ankle boot"]

x_train = x_train / 255.0
x_test = x_test / 255.0

# Visualize a few samples
plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(X_train[i], cmap='gray')
    plt.title(label_names[y_train[i]])
    plt.axis('off')
plt.tight_layout()
plt.show()
```

one hot encode

In [16]:
```
# Step 1 (continued): Flatten and normalize, then one-hot encode
X_train = X_train.reshape(-1, 28*28).astype('float32') / 255.0
X_test  = X_test.reshape(-1, 28*28).astype('float32') / 255.0

y_train = to_categorical(y_train, 10)
y_test  = to_categorical(y_test, 10)
```

Define and compile model

In [17]:
```
def create_model():
    model = Sequential([
        layers.Input(shape=(784,)),
        layers.Dense(512, activation='relu'),
        layers.Dropout(0.2),
```

```
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(10, activation='softmax')
    ])
    return model

model = create_model()
model.summary()

model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
dense_10 (Dense)	(None, 512)	401,920
dropout_6 (Dropout)	(None, 512)	0
dense_11 (Dense)	(None, 256)	131,328
dropout_7 (Dropout)	(None, 256)	0
dense_12 (Dense)	(None, 128)	32,896
dropout_8 (Dropout)	(None, 128)	0
dense_13 (Dense)	(None, 10)	1,290

Total params: 567,434 (2.16 MB)

Trainable params: 567,434 (2.16 MB)

Non-trainable params: 0 (0.00 B)

Train model

In [18]:

```
# Step 4: Train the model
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=20,
    batch_size=128,
    verbose=1
)
```

```
Epoch 1/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.7953 - loss: 0.5730 - val_acc
uracy: 0.8493 - val_loss: 0.4206
Epoch 2/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8526 - loss: 0.4085 - val_acc
uracy: 0.8581 - val_loss: 0.3988
Epoch 3/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8648 - loss: 0.3700 - val_acc
uracy: 0.8633 - val_loss: 0.3764
Epoch 4/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8751 - loss: 0.3446 - val_acc
uracy: 0.8710 - val_loss: 0.3596
Epoch 5/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8802 - loss: 0.3289 - val_acc
uracy: 0.8736 - val_loss: 0.3530
Epoch 6/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8837 - loss: 0.3150 - val_acc
uracy: 0.8800 - val_loss: 0.3338
Epoch 7/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8874 - loss: 0.3027 - val_acc
uracy: 0.8835 - val_loss: 0.3343
Epoch 8/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8921 - loss: 0.2918 - val_acc
uracy: 0.8766 - val_loss: 0.3395
Epoch 9/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8934 - loss: 0.2851 - val_acc
uracy: 0.8818 - val_loss: 0.3278
Epoch 10/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8978 - loss: 0.2742 - val_acc
uracy: 0.8823 - val_loss: 0.3413
Epoch 11/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8985 - loss: 0.2682 - val_acc
uracy: 0.8852 - val_loss: 0.3234
Epoch 12/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9007 - loss: 0.2629 - val_acc
uracy: 0.8868 - val_loss: 0.3256
Epoch 13/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9022 - loss: 0.2574 - val_acc
uracy: 0.8833 - val_loss: 0.3287
Epoch 14/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9038 - loss: 0.2545 - val_acc
uracy: 0.8890 - val_loss: 0.3173
Epoch 15/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step - accuracy: 0.9065 - loss: 0.2467 - val_acc
uracy: 0.8843 - val_loss: 0.3312
Epoch 16/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.9078 - loss: 0.2421 - val_acc
uracy: 0.8881 - val_loss: 0.3229
Epoch 17/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9108 - loss: 0.2362 - val_acc
uracy: 0.8889 - val_loss: 0.3169
Epoch 18/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9112 - loss: 0.2343 - val_acc
uracy: 0.8910 - val_loss: 0.3184
Epoch 19/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9125 - loss: 0.2298 - val_acc
```

```
uracy: 0.8901 - val_loss: 0.3249
Epoch 20/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9157 - loss: 0.2231 - val_acc
uracy: 0.8932 - val_loss: 0.3225
```

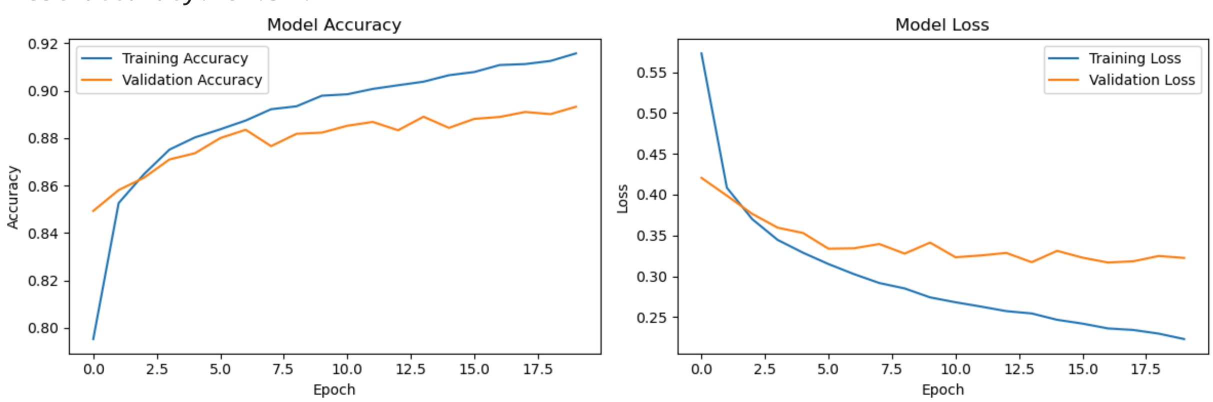
plot

In [19]:
```python
# Step 5: Evaluate test accuracy
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
print(f"Test accuracy: {test_acc*100:.2f}%")

# Plot accuracy and loss
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()
```

```
Test accuracy: 89.32%
```



Q2

Q2. For a given set of images with titles and without titles, which data set will have a better image classification performance? Explain and justify your answer.

The dataset with titles (labeled images) will lead to significantly better classification performance, because supervised learning relies on labeled data to compute error, adjust model weights, and improve predictive accuracy. Unlabeled images cannot train a classifier well without additional labeling or semi-supervised techniques.