```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
from sklearn.preprocessing import StandardScaler
from sklearn.inspection import DecisionBoundaryDisplay

# 1. 비선형 데이터 생성
# n_samples=100: 100개의 샘플
# noise=0.1: 데이터에 약간의 노이즈 추가
# random_state=0: 재현 가능성을 위해 난수 고정
X, y = datasets.make_moons(n_samples=100, noise=0.1, random_state=0)

# 2. 데이터 스케일링 (SVM은 스케일에 민감하므로 중요)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 3. 비교할 SVM 모델 정의
# C=1.0 : 정규화(Regularization) 파라미터
C = 1.0

models = (
    # (1) 선형 커널
    svm.SVC(kernel='linear', C=C),

    # (2) 다항 커널 (degree=3)
    # kernel='poly' : 여기서 커널 트릭이 적용됩니다.
    svm.SVC(kernel='poly', degree=3, C=C),

    # (3) RBF 커널
    # kernel='rbf' : 여기서 커널 트릭이 적용됩니다.
    # gamma='auto' : RBF 커널의 영향 범위를 자동으로 설정
    svm.SVC(kernel='rbf', gamma='auto', C=C)
)

# 모델 이름
titles = (
    'SVC with Linear kernel',
    'SVC with Polynomial (degree 3) kernel',
    'SVC with RBF kernel'
)

# 4. 모델 학습 및 시각화
# 1행 3열의 서브플롯 생성
fig, sub = plt.subplots(1, 3, figsize=(15, 5))

# meshgrid를 만들기 위해 X의 최소/최대 값 찾기
x_min, x_max = X_scaled[:, 0].min() - 1, X_scaled[:, 0].max() + 1
y_min, y_max = X_scaled[:, 1].min() - 1, X_scaled[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))
```

```
# 각 모델을 순회하며 학습하고 경계를 그림
for clf, title, ax in zip(models, titles, sub.flatten()):
    # 모델 학습
    clf.fit(X_scaled, y)

    # 결정 경계 시각화
    DecisionBoundaryDisplay.from_estimator(
        clf,
        np.c_[xx.ravel(), yy.ravel()],
        response_method="predict",
        cmap=plt.cm.coolwarm,
        alpha=0.8,
        ax=ax,
        xlabel=None,
        ylabel=None,
    )

    # 학습 데이터 포인트(산점도) 그리기
    ax.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y, cmap=plt.cm.coolwarm, s=20, edgecolors='k')
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)

plt.tight_layout()
plt.show()
```