DOMAIN NAME: CLOUD APPLICATION MANAGEMENT

PROJECT NAME: MACHINE LEARNING MODEL DEPLOYMENT WITH IBM CLOUD WATSON STUDIO

STARTING UP ML MODEL WITH WATSON STUDIO:

Deploying a machine learning model using IBM Cloud Watson Studio involves several steps. Here's a general outline to guide you:

1. Prepare the Model: Ensure your model is ready for deployment. This includes testing it thoroughly and optimizing it for production.

2. Create an IBM Cloud Account: If you haven't already, sign up for an IBM Cloud account to access the Watson Studio service.

3. Access Watson Studio: Once you're in the IBM Cloud dashboard, navigate to the Watson Studio service.

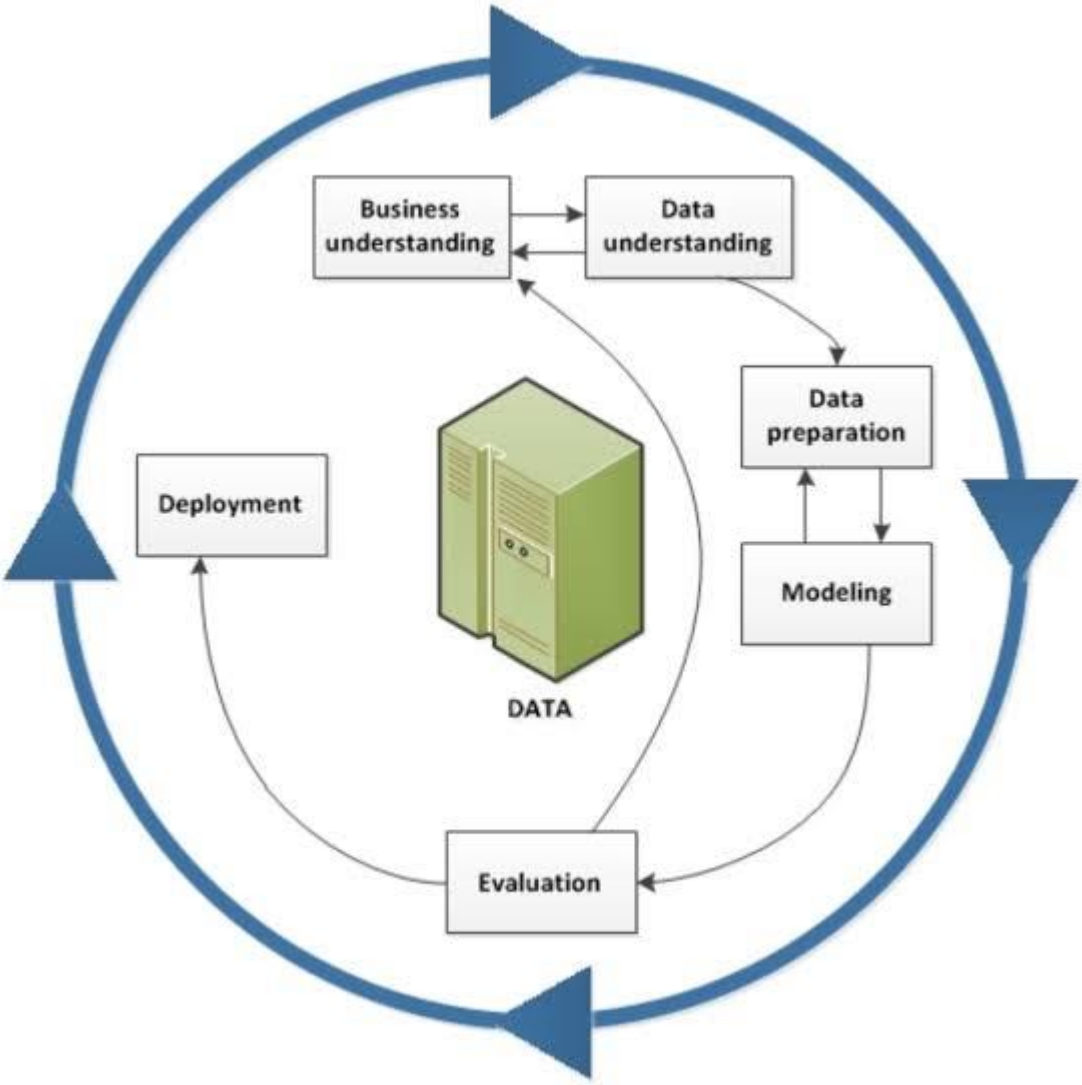
4. Create a Watson Studio Project: Set up a new project within Watson Studio. This will serve as the container for your model and related assets.

5. Add the Model to the Project: Upload your trained model to the Watson Studio project, ensuring it is compatible with the supported formats.

6. Deploy the Model: Within Watson Studio, you can use the tools provided to deploy your model. This typically involves configuring the deployment settings, selecting the desired infrastructure, and setting up the endpoint.

7. Monitor and Test: After deployment, monitor the model's performance and test it with sample data to ensure it's functioning as expected.

8. Integrate with Applications: Integrate the deployed model with your desired applications or services by utilizing the provided endpoint.



CODE SNIPPET:

Below is a basic code snippet for deploying a machine learning model using IBM Cloud Watson Studio:

python

Save the trained model

import joblib

joblib.dump(model, 'model.pkl')

Create a deployment space

```
from ibm_watson_machine_learning import APIClient

wml_credentials = {
   "url": "https://us-south.ml.cloud.ibm.com",
   "apikey": "YOUR_API_KEY"
}

client = APIClient(wml_credentials)
client.spaces.list()

# Deploy the model
metadata = {
   client.repository.ModelMetaNames.NAME: "My deployment",
   client.repository.ModelMetaNames.TYPE: "scikit-learn_0.23",
}

model_details = client.repository.store_model(model='model.pkl', meta_props=metadata)

model_uid = client.repository.get_model_uid(model_details)
model_deployment_details = client.deployments.create(
   artifact_uid=model_uid,
   name="My deployment"
)

deployment_uid = client.deployments.get_uid(model_deployment_details)
```

FRONTEND STRUCTURE:

```
html
<html>
<head>
```

```
  <title>Machine Learning Model Deployment</title>
</head>
<body>
  <h1>Machine Learning Model Deployment</h1>

  <p>Enter a value for the following feature:</p>
  <input type="text" id="feature_input" />

  <button type="button" onclick="predict()">Predict</button>

  <p id="prediction"></p>

  <script>
    function predict() {
      // Get the feature value from the input field.
      const feature_value = document.getElementById('feature_input').value;

      // Make a POST request to the backend API with the feature value.
      const request = new XMLHttpRequest();
      request.open('POST', '/predict');
      request.setRequestHeader('Content-Type', 'application/json');
      request.send(JSON.stringify({ feature_value }));

      // Handle the response from the backend API.
      request.onload = function() {
        if (request.status === 200) {
          // The prediction was successful.
          const prediction = JSON.parse(request.responseText).prediction;
          document.getElementById('prediction').innerHTML = `The prediction is:
${prediction}`;
        } else {
```

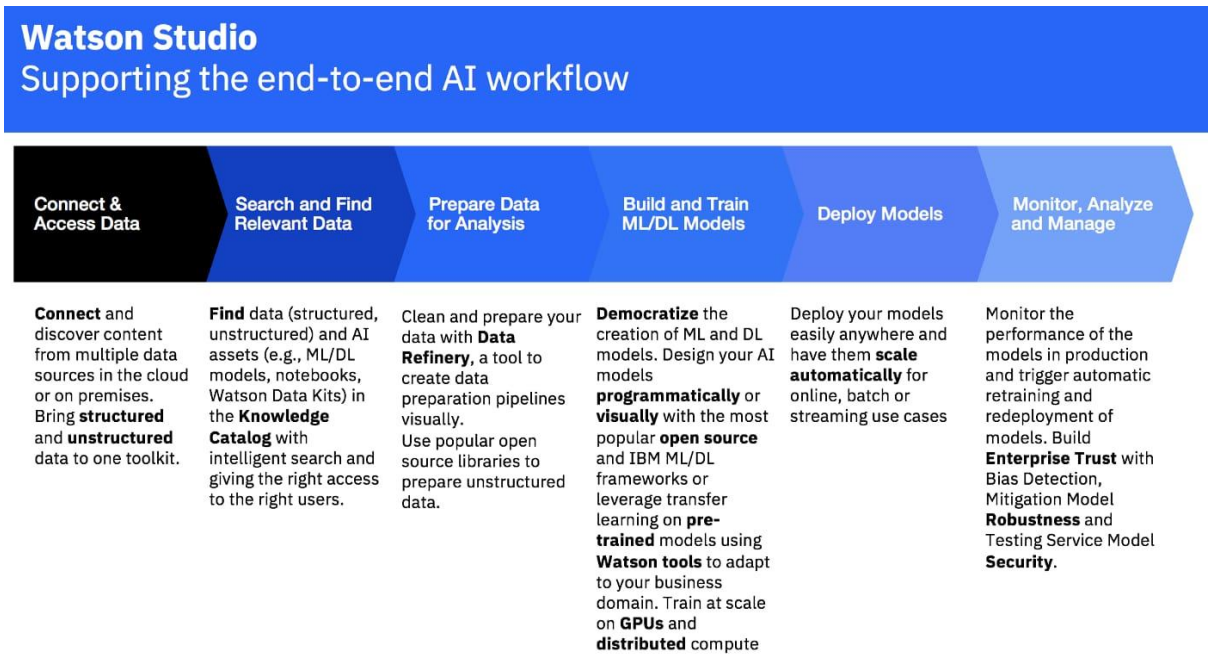
```
        // The prediction failed.
        document.getElementById('prediction').innerHTML = 'An error occurred while making the prediction.';
      }
    };
  }
 </script>
</body>
</html>
```

BACKEND STRUCTURE:

```
python
from flask import Flask, request, jsonify
import watson_machine_learning

# Create a Flask app.
app = Flask(__name__)

# Create a Watson Machine Learning client.
wml_client = watson_machine_learning.WatsonMachineLearningClient()

# Load the deployed machine learning model.
model = wml_client.load_model('my_model')

# Define a route to predict the outcome for a given feature value.
@app.route('/predict', methods=['POST'])
def predict():
  # Get the feature value from the request payload.
  feature_value = request.json['feature_value']
```

```
# Make a prediction using the deployed model.
prediction = model.predict(feature_value)

# Return the prediction to the client.
return jsonify({'prediction': prediction})

# Start the Flask app.
if __name__ == '__main__':
  app.run(debug=True)
```

SETTING UP ML MODEL WITH WATSON STUDIO:



To set up machine learning model deployment with IBM Cloud Watson Studio, you will need to:

1. Create an IBM Cloud account and Watson Studio service instance.
2. Build and train a machine learning model.
3. Deploy the model to a deployment space.
4. Monitor the deployed model.

Create an IBM Cloud account and Watson Studio service instance:

1. Go to the IBM Cloud website and create an account.
2. Once you have created an account, go to the Watson Studio catalog and create a service instance.
3. Select the region where you want to deploy your model.
4. Select the plan that meets your needs.
5. Click **Create.**

Build and train a machine learning model:

Watson Studio provides a variety of tools and resources for building and training machine learning models. You can use the visual modeling canvas, the notebook interface, or the Watson Machine Learning API.

Deploy the model to a deployment space:

Once you have built and trained a model, you can deploy it to a deployment space. Deployment spaces allow you to manage your deployments and make them available to your applications.

To deploy a model to a deployment space:

1. In Watson Studio, go to the **Assets** tab.
2. Select the model that you want to deploy.
3. Click **Deploy**.
4. Select the deployment space where you want to deploy the model.
5. Click **Deploy.**

Monitor the deployed model:

Once you have deployed a model, you should monitor its performance to ensure that it is meeting your needs. Watson Studio provides a variety of tools for monitoring deployed models, including:

- **Model accuracy:*** This metric measures how well the model is performing on new data.
- **Model latency:*** This metric measures how long it takes the model to generate a prediction.
- **Model drift:*** This metric measures how much the model's performance has changed over time.

You can use these metrics to identify any problems with your model and take corrective action, such as retraining the model or updating the data.

ADDITIONAL TIPS:

1. Before you deploy a model to production, it is important to test it thoroughly. You can use Watson Studio's model testing capabilities to test your model on a held-out dataset.

2. When you deploy a model to production, it is important to monitor its performance closely. You can use Watson Studio's model monitoring capabilities to monitor your model's accuracy, latency, and drift.

3. If you need to retrain your model, you can use Watson Studio's model retraining capabilities to retrain the model and deploy it to a new deployment space.