



NURTURING POTENTIAL

SAKET GYANPEETH'S  
**SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE**  
(Permanently Affiliated to University of Mumbai)

NAAC Accredited

Saket Vidyanagri Marg, Chinchpada Road, Katemanivali,  
Kalyan (East) -421306(Mah)

**Department of Information Technology**

This is to certify that

Mr./Ms. \_\_\_\_\_ Seat No. \_\_\_\_\_

of

**M.Sc. Information Technology**

**Part II NEP 2020 Semester III**

has satisfactorily carried out the required practical in the subject  
of \_\_\_\_\_

For the Academic year 2024 – 2025

---

**Practical In-Charge**

---

**Head of the Department**

---

**External Examiner**

**College Seal**



NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE

KALYAN (EAST)

ACADEMIC YEAR 2024-25

**M.Sc. Information Technology**

**Part II NEP 2020 Semester III**

**SUBMITTED BY**

---

**AS PRESCRIBED BY**

**UNIVERSITY OF MUMBAI**



**MUMBAI UNIVERSITY**

## **INDEX**

<b><u>Sr.No.</u></b>	<b><u>Practical Aim</u></b>	<b><u>Signature</u></b>
<b>1.</b>	Implementing advanced deep learning algorithms such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs) using Python libraries like TensorFlow or PyTorch.	
<b>2.</b>	Building a natural language processing (NLP) model for sentiment analysis or text classification.	
<b>3.</b>	Creating a chatbot using advanced techniques like transformer models.	
<b>4.</b>	Developing a recommendation system using collaborative filtering or deep learning approaches.	
<b>5.</b>	Implementing a computer vision project, such as object detection or image segmentation.	
<b>6.</b>	Training a generative adversarial network (GAN) for generating realistic images	
<b>7.</b>	Applying reinforcement learning algorithms to solve complex decision-making problems.	
<b>8.</b>	Utilizing transfer learning to improve model performance on limited datasets.	
<b>9.</b>	Building a deep learning model for time series forecasting or anomaly detection	
<b>10.</b>	Implementing a machine learning pipeline for automated feature engineering and model selection.	

## PRACTICAL – 1

**Aim:** Implementing advanced deep learning algorithms such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) using popular Python libraries like TensorFlow or PyTorch

### 1. Convolutional Neural Network (CNN) for Image Classification

CNNs are widely used in computer vision tasks, such as image classification, object detection, and segmentation. They are particularly effective for processing grid-like data (e.g., images) due to their ability to capture spatial hierarchies.

We'll implement a CNN for image classification on the **MNIST** dataset, a collection of handwritten digits.

#### Step 1: Install Dependencies

Install the required libraries if you don't have them:

```
pip install tensorflow matplotlib numpy
```

#### Step 2: Load and Preprocess the Dataset

We'll use the **MNIST** dataset, which is available directly in TensorFlow.

```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Normalize the images to [0, 1] and reshape them to (28, 28, 1)
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
x_train = np.expand_dims(x_train, axis=-1) # Add channel dimension
x_test = np.expand_dims(x_test, axis=-1) # Add channel dimension

# One-hot encode the labels
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

#### Step 3: Define the CNN Model

We will create a simple CNN architecture with 2 convolutional layers, followed by a fully connected (dense) layer.

```

def build_cnn_model():
    model = models.Sequential()

    # First convolutional layer
    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
    model.add(layers.MaxPooling2D((2, 2)))

    # Second convolutional layer
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))

    # Flatten the output for the dense layer
    model.add(layers.Flatten())

    # Fully connected layer
    model.add(layers.Dense(64, activation='relu'))

    # Output layer
    model.add(layers.Dense(10, activation='softmax')) # 10 classes for MNIST digits

    return model

# Build and compile the model
cnn_model = build_cnn_model()
cnn_model.compile(optimizer='adam',
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])

```

#### **Step 4: Train the Model**

Now, we'll train the CNN on the MNIST dataset.

```
# Train the CNN model
cnn_model.fit(x_train, y_train, epochs=5, batch_size=64, validation_split=0.1)
```

#### **Step 5: Evaluate the Model**

After training, we'll evaluate the model on the test set.

```
# Evaluate the model on the test data
test_loss, test_acc = cnn_model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_acc}')
```

#### **Step 6: Visualize the Results**

You can visualize the results by plotting the predictions made by the CNN.

```
# Make predictions
predictions = cnn_model.predict(x_test)

# Display the first 5 images and their predicted labels
for i in range(5):
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
    plt.title(f'Predicted: {np.argmax(predictions[i])}, Actual: {np.argmax(y_test[i])}')
    plt.show()
```

## 2. Recurrent Neural Network (RNN) for Sequence Prediction

RNNs are well-suited for sequential data like time-series, text, or audio. They maintain hidden states over time, making them effective for sequence modeling.

Let's implement a simple RNN using TensorFlow to predict the next word in a sequence.

### Step 1: Install Dependencies

If you don't have the required libraries yet, install them:

```
pip install tensorflow numpy
```

### Step 2: Prepare the Dataset

We'll use the **IMDB dataset** (a movie review dataset) for text classification. We'll build an RNN to predict the sentiment of a movie review (positive or negative).

```
# Load IMDB dataset for sentiment analysis
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Set maximum number of words to consider and maximum sequence length
max_features = 10000 # Only consider the top 10,000 words
 maxlen = 500 # Maximum length of the review

# Load and preprocess the dataset
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)

# Pad sequences to ensure they have the same length
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)
```

### Step 3: Define the RNN Model

We will define an RNN with **LSTM** layers. LSTM (Long Short-Term Memory) is an advanced type of RNN that solves the vanishing gradient problem, making it more effective for long sequences.

```
def build_rnn_model():
    model = models.Sequential()

    # Embedding layer to learn word representations
    model.add(layers.Embedding(input_dim=max_features, output_dim=128,
                               input_length=maxlen))

    # LSTM layer
    model.add(layers.LSTM(128))

    # Output layer (sigmoid for binary classification)
    model.add(layers.Dense(1, activation='sigmoid'))

    return model

# Build and compile the RNN model
rnn_model = build_rnn_model()
```

```
rnn_model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

### Step 4: Train the Model

We will train the RNN model on the IMDB dataset.

```
# Train the RNN model
rnn_model.fit(x_train, y_train, epochs=5, batch_size=64, validation_data=(x_test,
y_test))
```

### Step 5: Evaluate the Model

After training, we can evaluate the model's performance on the test data.

```
# Evaluate the model on the test data
test_loss, test_acc = rnn_model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_acc}')
```

### Step 6: Make Predictions

Once the model is trained, we can use it to make predictions on new reviews.

---

```
# Predict sentiment for the first review in the test set
predictions = rnn_model.predict(x_test[:5])

# Print predictions
for i, prediction in enumerate(predictions):
    sentiment = 'positive' if prediction >= 0.5 else 'negative'
    print(f"Review {i+1}: {sentiment} (probability: {prediction[0]:.2f})")
```

---

## Conclusion

In this we've demonstrated how to implement advanced deep learning algorithms using **TensorFlow**:

1. **CNN:** We used a convolutional neural network to classify handwritten digits from the MNIST dataset. CNNs are powerful for image recognition tasks, and they work by learning spatial hierarchies through convolution and pooling operations.
2. **RNN:** We used a recurrent neural network (RNN) with LSTM units to classify sentiment from movie reviews in the IMDB dataset. RNNs are ideal for sequence data because they maintain hidden states across time steps, enabling them to capture dependencies in the data.

## PRACTICAL – 2

### Aim: Building a Natural Language Processing (NLP) model for sentiment analysis or text classification

Sentiment analysis is a common Natural Language Processing (NLP) task that involves determining the sentiment or emotional tone of a text. This can be categorized as positive, negative, or neutral. Let's walk through a basic sentiment analysis example using Python and a popular NLP library, Huggingface Transformers.

We will use the **IMDb movie reviews dataset** to classify the sentiment of movie reviews as **positive** or **negative**. This dataset contains movie reviews labeled as 1 (positive) or 0 (negative), making it a binary classification task.

### Steps for Sentiment Analysis

#### 1. Install the required libraries:

```
pip install transformers datasets torch
```

- **transformers**: For using pre-trained models like BERT.
- **datasets**: A Huggingface library to load datasets like IMDb.
- **torch**: For deep learning with PyTorch.

#### 2. Import the libraries:

```
from datasets import load_dataset
from transformers import BertTokenizer, BertForSequenceClassification, Trainer,
TrainingArguments
import torch
```

#### 3. Load the IMDb dataset:

The `datasets` library from Huggingface makes it easy to load popular NLP datasets.

```
# Load the IMDb dataset
dataset = load_dataset('imdb')
print(dataset)
```

This loads the IMDb dataset and gives you the following splits:

- **train**: Training data
- **test**: Test data

Each review is a string of text, and each label is a sentiment (0 for negative, 1 for positive).

#### 4. Preprocess the data:

BERT requires tokenization, where the text is converted into token IDs that BERT can understand.

```

# Load the BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Tokenization function
def tokenize_function(examples):
    return tokenizer(examples['text'], padding="max_length", truncation=True)

# Apply tokenization to both the train and test data
train_data = dataset['train'].map(tokenize_function, batched=True)
test_data = dataset['test'].map(tokenize_function, batched=True)

# Set format for PyTorch
train_data.set_format(type='torch', columns=['input_ids', 'attention_mask',
                                             'label'])
test_data.set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])

```

Here, we:

- Load a pre-trained BERT tokenizer (`bert-base-uncased`), which converts text into tokens that the BERT model can understand.
- Define a `tokenize_function` that tokenizes the text and applies padding and truncation to ensure all sequences have the same length.
- Apply the tokenization function to both the training and testing datasets.
- Format the data into PyTorch tensors (`input_ids`, `attention_mask`, and `label`).

## 5. Load the pre-trained BERT model:

We now load the BERT model pre-trained on a large corpus of text and fine-tune it for sentiment analysis.

```
# Load the pre-trained BERT model for sequence classification (sentiment analysis)
model = BertForSequenceClassification.from_pretrained('bert-base-uncased',
                                                       num_labels=2)
```

`num_labels=2` indicates that the model will predict two possible classes (positive or negative).

## 6. Define Training Arguments:

The `TrainingArguments` class is used to specify how the model should be trained.

```

# Set up training arguments
training_args = TrainingArguments(
    output_dir='./results',                      # Output directory for model checkpoints
    evaluation_strategy="epoch",                  # Evaluate the model every epoch
    learning_rate=2e-5,                          # Learning rate for optimization
    per_device_train_batch_size=16,                # Batch size for training
    per_device_eval_batch_size=64,                 # Batch size for evaluation
    num_train_epochs=3,                           # Number of training epochs
    weight_decay=0.01,                            # L2 regularization
)

# Set up the Trainer
trainer = Trainer(
    model=model,                                 # The model to train
    args=training_args,                          # Training arguments
    train_dataset=train_data,                    # Training dataset
    eval_dataset=test_data,                      # Evaluation dataset
)
```

)

## 7. Train the Model:

Now we can start the training process.

```
# Train the model  
trainer.train()
```

The model will train for 3 epochs (as specified), during which it learns to classify the sentiment of reviews.

## 8. Evaluate the Model:

Once training is complete, we can evaluate how well the model performs on the test dataset.

```
# Evaluate the model  
results = trainer.evaluate()  
print(results)
```

The evaluation will return several metrics, including accuracy, precision, recall, and F1-score.

## 9. Predict Sentiment of New Reviews:

Now, let's make predictions on a new set of text data (movie reviews).

```
# Predict sentiment for a new review  
text = "I love this movie! It was amazing and the acting was superb."  
inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True,  
max_length=512)  
with torch.no_grad():  
    logits = model(**inputs).logits  
    prediction = torch.argmax(logits, dim=-1).item()  
  
# Convert the prediction to a sentiment label  
sentiment = "Positive" if prediction == 1 else "Negative"  
print(f"Sentiment: {sentiment}")
```

This code takes a new review (text), tokenizes it using the same tokenizer, and then feeds it into the trained BERT model to get the predicted sentiment.

## Conclusion:

In this example, we've used Huggingface's Transformers library to build a sentiment analysis model using the BERT model. The key steps include loading and tokenizing data, fine-tuning a pre-trained model, and evaluating its performance. This approach can be applied to many text classification tasks by choosing different datasets and pre-trained models.

## Overview of the Steps:

1. Data Loading
2. Data Preprocessing
3. Feature Extraction
4. Model Training

5. Model Evaluation
6. Prediction

We'll start by building a sentiment analysis model using Logistic Regression (Traditional Machine Learning model), and then use a deep learning-based model (BERT) for more advanced performance.

## Step 1: Data Loading

We'll use the **IMDb dataset** for this example. The dataset contains movie reviews labeled as positive or negative.

```
from datasets import load_dataset

# Load IMDb dataset
dataset = load_dataset('imdb')

# Inspect the data
print(dataset)
```

- The train set consists of 25,000 movie reviews, labeled as positive (1) or negative (0).
- The test set is similarly structured.

## Step 2: Data Preprocessing

We need to clean and tokenize the text before feeding it to the model. For traditional machine learning models, we can use **Bag of Words (BoW)** or **TF-IDF**. For deep learning models (like BERT), tokenization is handled by the model itself.

### Preprocessing for Logistic Regression (Traditional Machine Learning):

We'll use TF-IDF to transform text into numerical features.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# Use the 'text' column for the review and 'label' for sentiment
train_data = dataset['train']
X = train_data['text']
y = train_data['label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Use TF-IDF for feature extraction
tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

- **TF-IDF (Term Frequency-Inverse Document Frequency)** represents words based on their frequency in a document while considering the frequency across all documents, allowing the model to focus on important words.

### Preprocessing for BERT (Deep Learning Model):

BERT tokenization requires the Huggingface `transformers` library, which tokenizes text into subwords and converts them into IDs that the model can understand.

```
from transformers import BertTokenizer

# Load the BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Tokenize the text
def tokenize_function(examples):
    return tokenizer(examples['text'], padding=True, truncation=True,
max_length=512)

# Apply tokenization to both train and test datasets
train_data = dataset['train'].map(tokenize_function, batched=True)
test_data = dataset['test'].map(tokenize_function, batched=True)

# Convert datasets to PyTorch format
train_data.set_format(type='torch', columns=['input_ids', 'attention_mask',
'label'])
test_data.set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])
```

- **Tokenization:** Converts text into numerical token IDs. BERT uses subwords, meaning words are often broken into smaller pieces for more accurate representations.

### Step 3: Feature Extraction (TF-IDF vs. BERT)

- **TF-IDF** (Traditional Machine Learning Approach) creates a sparse matrix representing the text.
- **BERT** (Deep Learning Approach) uses embeddings that capture contextual meaning in text.

For traditional models, we use **TF-IDF**. For BERT, we directly use the tokenized inputs.

### Step 4: Model Training

#### Logistic Regression (Traditional Machine Learning Approach):

We'll train a Logistic Regression classifier on the TF-IDF features.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Initialize Logistic Regression model
lr_model = LogisticRegression(max_iter=100)

# Train the model
lr_model.fit(X_train_tfidf, y_train)

# Evaluate the model
y_pred = lr_model.predict(X_test_tfidf)
print(classification_report(y_test, y_pred))
```

Here we:

- Train a **Logistic Regression** model on the TF-IDF features.
- Evaluate the model using **classification metrics** (precision, recall, F1-score).

### BERT (Deep Learning Approach):

For BERT, we'll use the Trainer API from the **Huggingface Transformers** library to fine-tune a pre-trained BERT model.

```
from transformers import BertForSequenceClassification, Trainer, TrainingArguments

# Load the pre-trained BERT model
model = BertForSequenceClassification.from_pretrained('bert-base-uncased',
num_labels=2)

# Define training arguments
training_args = TrainingArguments(
    output_dir='./results',                                # Output directory for checkpoints
    evaluation_strategy="epoch",                           # Evaluate after each epoch
    learning_rate=2e-5,                                   # Learning rate
    per_device_train_batch_size=16,                        # Training batch size
    per_device_eval_batch_size=64,                          # Evaluation batch size
    num_train_epochs=3,                                    # Number of epochs
    weight_decay=0.01,                                     # Regularization
)

# Initialize Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_data,
    eval_dataset=test_data,
)

# Train the model
trainer.train()

# Evaluate the model
results = trainer.evaluate()
print(results)
```

- **BERT** is a deep learning model trained on large amounts of text data. We fine-tune it for sentiment analysis using the IMDb dataset.

### Step 5: Model Evaluation

For both models (Logistic Regression and BERT), we evaluate performance using metrics like **accuracy**, **precision**, **recall**, and **F1-score**.

For **Logistic Regression**, we use `classification_report` from **scikit-learn**. For **BERT**, we use the Trainer's built-in evaluation functionality.

## Step 6: Prediction

Once the model is trained, we can make predictions on new reviews:

### Logistic Regression (Traditional Model):

```
# Predict sentiment for a new review
new_review = ["This movie was fantastic! I loved the acting and the plot."]
new_review_tfidf = tfidf.transform(new_review)
predicted_sentiment = lr_model.predict(new_review_tfidf)

sentiment = "Positive" if predicted_sentiment == 1 else "Negative"
print(f"Sentiment: {sentiment}")
```

### BERT (Deep Learning Model):

```
# Tokenize the new review
inputs = tokenizer("This movie was fantastic! I loved the acting and the plot.",
return_tensors="pt", padding=True, truncation=True, max_length=512)

# Predict sentiment using the BERT model
with torch.no_grad():
    logits = model(**inputs).logits
    predicted_class = torch.argmax(logits, dim=-1).item()

sentiment = "Positive" if predicted_class == 1 else "Negative"
print(f"Sentiment: {sentiment}")
```

## Conclusion:

- **Traditional Machine Learning** (e.g., Logistic Regression with TF-IDF) is faster and easier to implement but may not capture deep contextual information in text.
- **Deep Learning Models** (e.g., BERT) are more powerful and capture complex relationships and context in text, but they require more computational resources and time for training.

Both approaches are effective for sentiment analysis, but the choice between them depends on the specific use case, available resources, and the complexity of the task.

**PRACTICAL – 3**

**Aim:** Creating a chatbot using advanced techniques like the Transformer models.

**Key Steps to Build a Transformer-Based Chatbot:**

1. **Choose a Pre-trained Transformer Model:** You can use models like **GPT**, **DialoGPT**, or **BERT-based models** fine-tuned for conversational tasks.
2. **Set Up the Development Environment:** Install the required libraries.
3. **Data Preprocessing:** Although pre-trained models are already well-suited for many tasks, data can be fine-tuned for better domain-specific performance.
4. **Build the Chatbot Interface:** Set up the chatbot interface to communicate with the model.
5. **Deploy the Model:** You can run the model locally or deploy it on a cloud platform.

**Step 1: Install Required Libraries**

First, we need to install the **Huggingface Transformers** library or the **OpenAI GPT API**. Here's how to set up each.

**Option 1: Huggingface (DialoGPT)**

```
pip install transformers torch
```

**Option 2: OpenAI API (for GPT-3/4)**

For GPT-3/4 (via OpenAI's API), you first need an API key from OpenAI.

```
pip install openai
```

**Step 2: Build a Simple Chatbot with DialoGPT (Huggingface)**

DialoGPT is a variant of GPT-2 fine-tuned for conversation. We can use it to build an interactive chatbot.

**1. Import the Libraries**

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch
```

**2. Load the Pre-trained Model and Tokenizer**

```
# Load DialoGPT model and tokenizer
model_name = "microsoft/DialoGPT-medium"
model = AutoModelForCausalLM.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

- **DialoGPT-medium** is a smaller version of the model. You can also try **DialoGPT-large** if you need more power.

### 3. Create a Chat Function

```
def chat_with_bot(input_text, chat_history=None):
    # Encode the new user input, add the eos_token and return a tensor in Pytorch
    new_user_input_ids = tokenizer.encode(input_text + tokenizer.eos_token,
    return_tensors='pt')

    # Append the new user input tokens to the chat history (if exists)
    bot_input_ids = new_user_input_ids if chat_history is None else
    torch.cat([chat_history, new_user_input_ids], dim=-1)

    # Generate a response from the model
    chat_history = model.generate(bot_input_ids, max_length=1000,
    pad_token_id=tokenizer.eos_token_id, no_repeat_ngram_size=3, top_p=0.92,
    temperature=0.75)

    # Decode the generated response
    bot_output = tokenizer.decode(chat_history[:, bot_input_ids.shape[-1]:][0],
    skip_special_tokens=True)

    return bot_output, chat_history
```

- **chat\_with\_bot()**: This function takes the user input and returns the model's response while maintaining the conversation history.
- **max\_length**: Maximum number of tokens for the generated output.
- **top\_p and temperature**: Control the creativity of the response. Lower values (e.g., 0.75) make the model more deterministic.

### 4. Interact with the Chatbot

```
chat_history = None

while True:
    user_input = input("You: ")
    if user_input.lower() == "quit":
        break

    bot_response, chat_history = chat_with_bot(user_input, chat_history)
    print(f"Bot: {bot_response}")
```

This simple loop allows the user to interact with the chatbot in a conversational manner. Type "**quit**" to end the conversation.

### Step 3: Build a Chatbot with GPT-3/4 (OpenAI API)

If you prefer to use **GPT-3/4**, which is very advanced and capable of handling more complex conversations, here's how you can do it using OpenAI's API.

#### 1. Set Up OpenAI API Key

First, set your OpenAI API key. If you haven't gotten one, sign up at [OpenAI's platform](#).

```
import openai

# Set up the OpenAI API key
openai.api_key = 'your-api-key-here'
```

## 2. Create a Function for Chatbot Interaction

```
def chat_with_gpt3(input_text):
    response = openai.Completion.create(
        engine="text-davinci-003", # You can use "gpt-3.5-turbo" or "gpt-4"
        depending on your API access
        prompt=input_text,
        max_tokens=150,
        temperature=0.7,
        top_p=1.0,
        frequency_penalty=0.0,
        presence_penalty=0.0,
        stop=["\n"]
    )

    return response.choices[0].text.strip()
```

- **engine:** Choose the model you want to use (e.g., davinci, gpt-3.5-turbo, gpt-4).
- **temperature:** Controls the randomness of the response (0.7 is a good middle ground).

## 3. Chat with GPT-3

```
while True:
    user_input = input("You: ")
    if user_input.lower() == "quit":
        break

    bot_response = chat_with_gpt3(user_input)
    print(f"Bot: {bot_response}")
```

## Step 4: Improve the Chatbot with Memory (Optional)

### Example (Huggingface DialoGPT with memory):

```
def chat_with_memory(input_text, chat_history=None):
    # Encode the user input and concatenate with previous chat history
    new_user_input_ids = tokenizer.encode(input_text + tokenizer.eos_token,
    return_tensors='pt')
    bot_input_ids = new_user_input_ids if chat_history is None else
    torch.cat([chat_history, new_user_input_ids], dim=-1)

    # Generate response
    chat_history = model.generate(bot_input_ids, max_length=1000,
    pad_token_id=tokenizer.eos_token_id, no_repeat_ngram_size=3, top_p=0.92,
    temperature=0.75)
    bot_output = tokenizer.decode(chat_history[:, bot_input_ids.shape[-1]:][0],
    skip_special_tokens=True)

    return bot_output, chat_history
```

### Step 5: Deploy the Chatbot (Optional)

After developing your chatbot, you can deploy it to various platforms:

1. **Web Deployment:** Use **Flask** or **FastAPI** to build a web API for your chatbot.
2. **Messaging Platforms:** Integrate the chatbot into platforms like **Slack**, **Discord**, or **Telegram** using their respective bot APIs.

### Conclusion

By leveraging pre-trained Transformer models like **DialoGPT** or **GPT-3/4**, you can quickly build a powerful and scalable chatbot. These models can understand context, generate human-like responses, and be fine-tuned for specific tasks or domains. Whether you use **Huggingface Transformers** or the **OpenAI API**, both approaches allow you to build state-of-the-art conversational agents.

**PRACTICAL – 4**

**Aim: Developing a recommendation system using collaborative filtering or deep learning approaches.**

1. **Collaborative Filtering** (a traditional method)
2. **Deep Learning Approaches** (using neural networks)

## 1. Collaborative Filtering

Collaborative filtering is based on the idea that users who agreed in the past will agree in the future. It predicts a user's preferences based on the preferences of other similar users.

There are two types of collaborative filtering:

- **User-based Collaborative Filtering:** Recommends items by finding similar users to the target user and recommending items those similar users liked.
- **Item-based Collaborative Filtering:** Recommends items similar to the items the user has already liked.

In this example, we'll focus on **Matrix Factorization**, which is a popular collaborative filtering technique.

### Step 1: Install Libraries

```
pip install numpy pandas scikit-learn surprise
```

The **surprise** library is a Python library that implements collaborative filtering and matrix factorization methods.

### Step 2: Prepare the Data

We'll use a **movie ratings dataset** as an example. You can use a dataset like **MovieLens**.

```
from surprise import Dataset, Reader
import pandas as pd

# Load the MovieLens dataset (example with ratings)
url =
"https://raw.githubusercontent.com/sidooms/MovieTweetings/master/latest/ratings.dat"
ratings = pd.read_csv(url, sep="::", header=None, names=["user_id", "item_id",
"rating", "timestamp"])

# Prepare the data for surprise
reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(ratings[['user_id', 'item_id', 'rating']], reader)
```

- **ratings** contains user-item interactions, such as movie ratings.
- **Dataset.load\_from\_df()** prepares the data for use in the collaborative filtering model.

### Step 3: Apply Collaborative Filtering using Matrix Factorization (SVD)

```
from surprise import SVD
from surprise.model_selection import train_test_split
from surprise import accuracy

# Split the dataset into train and test
trainset, testset = train_test_split(data, test_size=0.2)

# Use Singular Value Decomposition (SVD)
svd = SVD()

# Train the model
svd.fit(trainset)

# Test the model
predictions = svd.test(testset)

# Evaluate the accuracy
accuracy.rmse(predictions)
```

- **SVD (Singular Value Decomposition):** A matrix factorization technique used in collaborative filtering. It decomposes the user-item interaction matrix into latent factors, which represent hidden relationships between users and items.

### Step 4: Make Recommendations

To make a recommendation for a specific user, we can predict ratings for all items and suggest the ones with the highest predicted ratings.

```
def recommend(user_id, n=10):
    # Get a list of all item IDs
    all_items = ratings['item_id'].unique()

    # Predict ratings for each item
    predictions = [svd.predict(user_id, item_id) for item_id in all_items]

    # Sort predictions by rating (descending order)
    sorted_predictions = sorted(predictions, key=lambda x: x.est, reverse=True)

    # Get the top n recommended items
    top_n = sorted_predictions[:n]
    return [(pred.iid, pred.est) for pred in top_n]

# Example: Recommend 10 items for user 1
recommendations = recommend(user_id=1, n=10)
print(recommendations)
```

## 2. Deep Learning Approaches for Recommendation Systems

Deep learning-based recommendation systems aim to use more complex models, such as neural networks, to capture hidden patterns in user-item interactions. One popular architecture is the **Autoencoder** or a **Neural Collaborative Filtering (NCF)** model.

## Step 1: Install Required Libraries

```
pip install tensorflow numpy pandas scikit-learn
```

## Step 2: Data Preparation

You can use the same dataset as in the collaborative filtering example. We will create a user-item matrix where each row represents a user, and each column represents an item. The values will be the ratings.

```
import numpy as np

# Create user-item matrix
user_item_matrix = ratings.pivot(index='user_id', columns='item_id',
values='rating').fillna(0)
```

- **pivot()**: This creates a matrix where rows represent users, columns represent items, and values are the ratings.

## Step 3: Build the Neural Network Model (Autoencoder)

```
import tensorflow as tf
from tensorflow.keras import layers, models

# Define the model architecture (Autoencoder)
def create_model(input_dim, encoding_dim):
    # Input layer
    input_layer = layers.Input(shape=(input_dim,))

    # Encoder
    encoded = layers.Dense(encoding_dim, activation='relu')(input_layer)

    # Decoder
    decoded = layers.Dense(input_dim, activation='sigmoid')(encoded)

    # Autoencoder model
    autoencoder = models.Model(input_layer, decoded)

    # Encoder model (to extract the compressed features)
    encoder = models.Model(input_layer, encoded)

    # Compile the model
    autoencoder.compile(optimizer='adam', loss='mean_squared_error')

    return autoencoder, encoder

# Set parameters
input_dim = user_item_matrix.shape[1] # Number of items
encoding_dim = 100 # Compressed representation

# Create the autoencoder model
autoencoder, encoder = create_model(input_dim, encoding_dim)

# Train the autoencoder
autoencoder.fit(user_item_matrix.values, user_item_matrix.values, epochs=50,
batch_size=256, shuffle=True, validation_data=(user_item_matrix.values,
user_item_matrix.values))
```

**Autoencoder:** An unsupervised neural network used for dimensionality reduction. The encoder part compresses the input (user-item matrix) into a lower-dimensional representation, while the decoder reconstructs the original input.

#### Step 4: Make Recommendations

Once the autoencoder is trained, we can generate recommendations for users by using the encoder to get the compressed representation of the user-item matrix.

```
# Get the compressed user-item matrix from the encoder
encoded_matrix = encoder.predict(user_item_matrix.values)

# Recommend items for a user (use the reconstruction error as a measure)
def recommend_deep(user_id, n=10):
    user_encoded = encoded_matrix[user_id - 1] # Get the encoding for the specific user
    predictions = np.dot(user_encoded, encoded_matrix.T) # Calculate predicted ratings for each item

    # Sort by predicted ratings (descending)
    recommended_items = np.argsort(predictions)[-n:][::-1]

    return recommended_items

# Example: Recommend 10 items for user 1
recommended_items = recommend_deep(user_id=1, n=10)
print("Recommended items for user 1:", recommended_items)
```

- **Autoencoder-based Recommendations:** We use the encoder to obtain compressed user-item representations and then calculate the dot product between the compressed user vector and all other item vectors to predict ratings.

#### Conclusion

Both **Collaborative Filtering** and **Deep Learning Approaches** are widely used for building recommendation systems, and each has its strengths:

- **Collaborative Filtering** is a traditional method based on user-item interactions. It works well for small datasets and has simpler models (e.g., matrix factorization with SVD).
- **Deep Learning Approaches** (e.g., Autoencoders, Neural Collaborative Filtering) are more advanced and can capture complex relationships in the data, making them more suitable for large datasets with more intricate patterns.

PRACTICAL – 5

**Aim: Implementing a Computer Vision project, such as Object Detection or Image Segmentation.**

### Project 1: Object Detection using TensorFlow and OpenCV

**Object detection** involves detecting instances of objects within an image and classifying them. The object detection model will output the locations (bounding boxes) of these objects as well as their labels (e.g., person, car, etc.).

We will use **TensorFlow's Object Detection API** to implement object detection. This library provides pre-trained models for different object detection tasks.

#### Step 1: Install Dependencies

First, install the required libraries.

```
pip install tensorflow opencv-python opencv-python-headless
```

Additionally, we will install the **TensorFlow Object Detection API**.

```
pip install tf-slim
pip install tensorflow-object-detection-api
```

You also need to install the following dependencies to run the Object Detection API:

```
pip install --upgrade setuptools
pip install pycocotools
```

#### Step 2: Load Pre-trained Model

TensorFlow provides pre-trained models, such as **Faster R-CNN**, **SSD**, and **YOLO**, for object detection. For simplicity, we'll use the **SSD MobileNet V2** model, which is fast and performs well.

```
import tensorflow as tf
import numpy as np
import os
import cv2
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util

# Load pre-trained model
PATH_TO_CKPT = 'ssd_mobilenet_v2_coco_2018_03_29/frozen_inference_graph.pb'
model = tf.saved_model.load(PATH_TO_CKPT)

# Load label map (for COCO dataset)
PATH_TO_LABELS = 'mscoco_label_map.pbtxt'
category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS,
use_display_name=True)
```

- `ssd_mobilenet_v2_coco_2018_03_29/frozen_inference_graph.pb` is a frozen model file (you can download it from the TensorFlow model zoo).
- `mscoco_label_map.pbtxt` is the label map that corresponds to the COCO dataset classes.

### Step 3: Prepare the Image

Now, we'll prepare the input image and run it through the model.

```
# Load an image
image_path = 'image.jpg' # Path to input image
image_np = cv2.imread(image_path)

# Convert image to RGB (OpenCV loads in BGR by default)
image_rgb = cv2.cvtColor(image_np, cv2.COLOR_BGR2RGB)

# Convert to a tensor
input_tensor = tf.convert_to_tensor(image_rgb)
input_tensor = input_tensor[tf.newaxis,...] # Add batch dimension
```

### Step 4: Perform Object Detection

We now perform object detection on the input image.

```
# Run detection
model_fn = model.signatures['serving_default']
output_dict = model_fn(input_tensor)

# The model outputs various results:
# 'num_detections', 'detection_boxes', 'detection_scores', 'detection_classes'
num_detections = int(output_dict['num_detections'][0])
detection_boxes = output_dict['detection_boxes'][0].numpy()
detection_scores = output_dict['detection_scores'][0].numpy()
detection_classes = output_dict['detection_classes'][0].numpy().astype(np.int32)
```

- `detection_boxes`: Bounding boxes for each object detected.
- `detection_scores`: Confidence scores for each object detected.
- `detection_classes`: Class IDs of detected objects.

### Step 5: Visualize the Results

We can visualize the detected objects using OpenCV and TensorFlow's visualization utilities.

```
# Visualize detected boxes and labels on the image
vis_util.visualize_boxes_and_labels_on_image_array(
    image_np,
    detection_boxes,
    detection_classes,
    detection_scores,
    category_index,
    instance_masks=None,
    use_normalized_coordinates=True,
    line_thickness=8
)
```

```
# Convert image back to BGR for OpenCV display
image_bgr = cv2.cvtColor(image_np, cv2.COLOR_RGB2BGR)

# Display the output image
cv2.imshow('Object Detection', image_bgr)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- `visualize_boxes_and_labels_on_image_array()` draws the bounding boxes on the image, along with labels and confidence scores.

## Step 6: Saving the Result

You can also save the resulting image with the detected objects.

```
# Save the output image
cv2.imwrite('output_image.jpg', image_bgr)
```

## Project 2: Image Segmentation using U-Net

**Image segmentation** is the task of classifying each pixel in an image as belonging to a specific class (e.g., background, object, etc.). In this example, we'll use a **U-Net** architecture, which is widely used for segmentation tasks.

We'll use the **Keras** library to implement U-Net for semantic segmentation.

### Step 1: Install Dependencies

If you don't have **Keras** or **TensorFlow** installed:

```
pip install tensorflow
```

### Step 2: Build the U-Net Model

Here's a simplified U-Net model built using **Keras**.

```
import tensorflow as tf
from tensorflow.keras import layers, models

# Build the U-Net model
def unet_model(input_size=(256, 256, 3)):
    inputs = layers.Input(input_size)

    # Encoder
    conv1 = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)
    conv1 = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(conv1)
    pool1 = layers.MaxPooling2D((2, 2))(conv1)

    # Decoder
    up1 = layers.UpSampling2D((2, 2))(pool1)
    concat1 = layers.concatenate([conv1, up1], axis=-1)
    conv2 = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(concat1)

    # Output layer
    outputs = layers.Conv2D(1, (1, 1), activation='sigmoid')(conv2)

    return models.Model(inputs=inputs, outputs=outputs)
```

```

model = models.Model(inputs, outputs)
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

return model

# Create the U-Net model
model = unet_model()
model.summary()

```

- **Conv2D**: Convolutional layers for feature extraction.
- **MaxPooling2D**: Downsampling the feature map.
- **UpSampling2D**: Upsampling to reconstruct the segmentation map.
- **concatenate**: Skip connections to preserve spatial information.

### Step 3: Train the Model

Now, train the model with a dataset. We can use any segmentation dataset (for example, **Pascal VOC** or **COCO**), or you can create a simple dataset with images and corresponding masks (ground truth segmentation maps).

```

# Train the model
# X_train: input images, Y_train: segmentation masks
model.fit(X_train, Y_train, batch_size=16, epochs=10)

```

### Step 4: Make Predictions

Once the model is trained, you can use it to predict segmentation masks for new images.

```

# Predict segmentation map for a new image
segmentation_map = model.predict(X_test)

# Display the result
import matplotlib.pyplot as plt
plt.imshow(segmentation_map[0, :, :, 0], cmap='gray')
plt.show()

```

### Step 5: Post-Processing and Visualization

You can further process and visualize the segmentation map by overlaying it on the original image.

```

# Threshold segmentation map to get a binary mask
binary_mask = (segmentation_map[0, :, :, 0] > 0.5).astype(np.uint8)

# Overlay the mask on the original image
segmented_image = cv2.addWeighted(X_test[0], 0.7, binary_mask * 255, 0.3, 0)

# Display the segmented image
plt.imshow(segmented_image)
plt.show()

```

## Conclusion

- **Object Detection** was implemented using TensorFlow's Object Detection API.
- **Image Segmentation** was implemented using a simple **U-Net** model in Keras.

**PRACTICAL – 6****Aim: Training a Generative Adversarial Network (GAN) for Generating Realistic Images**

**Generative Adversarial Networks (GANs)** are a class of machine learning models used to generate synthetic data, including realistic images. A GAN consists of two networks: a **generator** and a **discriminator**. The generator creates images, and the discriminator tries to differentiate between real images (from the dataset) and fake images (from the generator). The generator aims to fool the discriminator, and through this adversarial process, both models improve.

**Steps Involved in Training a GAN**

1. **Prepare the dataset:** Load and preprocess the dataset.
2. **Build the GAN architecture:** Create the generator and discriminator networks.
3. **Define the loss functions:** Use binary cross-entropy for both networks.
4. **Train the GAN:** Train both networks in an adversarial manner.
5. **Generate images:** Use the trained generator to produce images.

**Step-by-Step Implementation****Step 1: Install Dependencies**

First, we need to install the required libraries.

```
pip install tensorflow matplotlib numpy
```

**Step 2: Import Libraries**

We will import the necessary libraries for building and training the GAN.

```
import tensorflow as tf
from tensorflow.keras import layers
import numpy as np
import matplotlib.pyplot as plt
```

**Step 3: Load and Preprocess the Dataset**

We'll use the **MNIST** dataset, which consists of 28x28 grayscale images of handwritten digits (0-9).

```
# Load MNIST dataset
(X_train, _), (_, _) = tf.keras.datasets.mnist.load_data()

# Normalize the images to [-1, 1]
X_train = X_train.astype("float32")
X_train = (X_train - 127.5) / 127.5 # Normalize to the range [-1, 1]
X_train = np.expand_dims(X_train, axis=-1) # Add a channel dimension
```

- **Normalization:** This is necessary because GANs are more stable when inputs range between [-1, 1].

## Step 4: Define the Generator and Discriminator

### Generator Model

The **Generator** takes random noise as input and generates images from it. We will use several **dense** and **conv2d** layers to upsample the noise into an image.

```
def build_generator():
    model = tf.keras.Sequential()
    model.add(layers.InputLayer(input_shape=(100,))) # Random noise of size 100

    model.add(layers.Dense(7 * 7 * 256, use_bias=False)) # Dense layer
    model.add(layers.BatchNormalization()) # Batch normalization
    model.add(layers.LeakyReLU()) # Leaky ReLU activation

    model.add(layers.Reshape((7, 7, 256))) # Reshape to a 7x7x256 tensor
    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same',
use_bias=False))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same',
use_bias=False))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same',
use_bias=False, activation='tanh')) # Output layer

    return model

generator = build_generator()
```

- **Dense Layer:** Starts with a fully connected layer that takes a 100-dimensional vector (noise) as input and outputs a flattened 7x7x256 tensor.
- **Conv2DTranspose:** Upsamples the data, converting it into a higher resolution image.
- **LeakyReLU:** Used for non-linear activation.

### Discriminator Model

The **Discriminator** takes an image as input and outputs a single scalar value (0 or 1), indicating whether the image is real (from the dataset) or fake (generated by the generator).

```
def build_discriminator():
    model = tf.keras.Sequential()
    model.add(layers.InputLayer(input_shape=(28, 28, 1)))

    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3)) # Dropout for regularization

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten()) # Flatten to 1D vector
    model.add(layers.Dense(1, activation='sigmoid')) # Output layer: real or fake
```

```

    return model

discriminator = build_discriminator()

```

- **Conv2D:** Convolution layers are used to extract features from the image.
- **LeakyReLU:** Activation function that allows small negative values.
- **Dropout:** Used for regularization to prevent overfitting.

## Step 5: Define the GAN Model

The **GAN** model is a combination of the generator and discriminator. The generator's goal is to produce images that can fool the discriminator into classifying them as real.

```

# GAN model: The generator produces images, and the discriminator classifies them
discriminator.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# The discriminator's weights are frozen during the training of the GAN
discriminator.trainable = False

gan_input = layers.Input(shape=(100,))
x = generator(gan_input)
gan_output = discriminator(x)

gan = tf.keras.Model(gan_input, gan_output)
gan.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

```

- **Frozen Discriminator:** During the GAN training, only the generator is trained. The discriminator's weights are kept frozen to prevent it from being updated during this phase.

## Step 6: Training the GAN

We now set up the training loop. The GAN training alternates between training the discriminator and training the generator.

```

import time

def train_gan(epochs, batch_size):
    # Real labels are 1 and fake labels are 0
    real_labels = np.ones((batch_size, 1))
    fake_labels = np.zeros((batch_size, 1))

    for epoch in range(epochs):
        start_time = time.time()

        # Train the discriminator on real images
        idx = np.random.randint(0, X_train.shape[0], batch_size)
        real_images = X_train[idx]
        d_loss_real = discriminator.train_on_batch(real_images, real_labels)

        # Train the discriminator on fake images generated by the generator
        noise = np.random.randn(batch_size, 100)
        fake_images = generator.predict(noise)
        d_loss_fake = discriminator.train_on_batch(fake_images, fake_labels)

        # Calculate the total discriminator loss
        d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

```

```

# Train the generator through the GAN model
g_loss = gan.train_on_batch(noise, real_labels)

# Print progress
elapsed_time = time.time() - start_time
print(f'{epoch}/{epochs} | D Loss: {d_loss[0]} | G Loss: {g_loss[0]} | Time: {elapsed_time:.2f}s')

# Save generated images periodically
if epoch % 1000 == 0:
    save_generated_images(epoch)

def save_generated_images(epoch, examples=16, dim=(4, 4), figsize=(4, 4)):
    noise = np.random.randn(examples, 100)
    generated_images = generator.predict(noise)

    plt.figure(figsize=figsize)
    for i in range(examples):
        plt.subplot(dim[0], dim[1], i+1)
        plt.imshow(generated_images[i, :, :, 0], cmap='gray')
        plt.axis('off')
    plt.tight_layout()
    plt.savefig(f'generated_image_epoch_{epoch}.png')
    plt.close()

# Train the GAN
train_gan(epochs=10000, batch_size=64)

```

- **Training Loop:**
  - **Discriminator:** Trains on both real and fake images.
  - **Generator:** Trains via the GAN model to fool the discriminator into classifying fake images as real.
- **Image Saving:** We periodically save generated images to visualize how the model is improving.

## Step 7: Visualize the Results

During training, the generator improves its ability to produce realistic images. After training, you can generate new images.

```

# Generate and visualize new images after training
noise = np.random.randn(16, 100)
generated_images = generator.predict(noise)

plt.figure(figsize=(4, 4))
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(generated_images[i, :, :, 0], cmap='gray')
    plt.axis('off')
plt.tight_layout()
plt.show()

```

## Conclusion

We've successfully implemented and trained a **Deep Convolutional GAN (DCGAN)** using TensorFlow for generating realistic images. The **generator** creates fake images, and the **discriminator** attempts to classify them as real or fake.

PRACTICAL – 7**Aim: Applying Reinforcement Learning (RL) Algorithms to Solve Complex Decision-Making Problems**

Reinforcement Learning (RL) is a type of machine learning where an agent learns how to make decisions by interacting with an environment. The goal is to learn an optimal policy that maximizes cumulative rewards over time. In RL, an agent takes actions in an environment and receives feedback in the form of rewards or penalties, allowing it to adjust its actions accordingly.

**Problem: Solving a Grid World Problem with Q-Learning**

A **Grid World** is a simple environment where an agent moves around a grid to reach a goal. The grid can have obstacles, and the agent must learn to navigate it to maximize rewards (for example, reaching the goal in the fewest steps).

We'll demonstrate the application of **Q-Learning** in a simple grid world setup.

**Steps:**

1. **Set up the environment:** Define the grid world with rewards, states, and actions.
2. **Define the Q-learning algorithm:** Define how the agent learns the optimal policy.
3. **Train the agent:** Let the agent explore the environment and update its Q-values.
4. **Evaluate the learned policy:** Test the agent after training.

**Step-by-Step Implementation****Step 1: Install Required Libraries**

You need **NumPy** for array manipulation. Install it if you don't have it.

```
pip install numpy
```

**Step 2: Set Up the Environment**

We will create a grid of size 5x5. The agent starts at the top-left corner (0,0) and must reach the goal at the bottom-right corner (4,4).

```
import numpy as np

# Define the grid world
grid_size = 5 # 5x5 grid
goal = (4, 4) # Goal position
obstacles = [(1, 1), (2, 1), (3, 3)] # Obstacles that the agent cannot pass

# Define the rewards for each cell
reward_grid = np.zeros((grid_size, grid_size))
reward_grid[goal] = 1 # Positive reward at the goal
for obs in obstacles:
    reward_grid[obs] = -1 # Negative reward for obstacles
```

```
# Define actions: Up, Down, Left, Right
actions = [(0, -1), (0, 1), (-1, 0), (1, 0)] # (dy, dx) for up, down, left, right

# Helper function to check if a position is within the grid and not an obstacle
def is_valid_move(pos):
    if 0 <= pos[0] < grid_size and 0 <= pos[1] < grid_size and pos not in obstacles:
        return True
    return False
```

### Step 3: Q-Learning Algorithm

Q-learning is an off-policy RL algorithm. It learns the value of state-action pairs (Q-values), and the policy is derived by selecting the action with the highest Q-value in each state.

The Q-learning update rule is:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Where:

- $Q(s,a)$  is the Q-value for state  $s$  and action  $a$ ,
- $\alpha$  is the learning rate,
- $\gamma$  is the discount factor,
- $r$  is the reward for taking action  $a$  from state  $s$ ,
- $s'$  is the new state after taking action  $a$ ,
- $a'$  is the next action chosen.

```
# Initialize Q-table with zeros
Q = np.zeros((grid_size, grid_size, len(actions))) # Q(s, a)

# Hyperparameters
alpha = 0.1 # Learning rate
gamma = 0.9 # Discount factor
epsilon = 0.1 # Exploration rate
episodes = 1000 # Number of training episodes
max_steps = 100 # Maximum steps per episode

# Training the agent
def train_q_learning():
    for episode in range(episodes):
        state = (0, 0) # Start at the top-left corner
        total_reward = 0

        for step in range(max_steps):
            # Exploration vs. Exploitation
            if np.random.rand() < epsilon:
                action_idx = np.random.choice(len(actions)) # Random action
(exploration)
            else:
                action_idx = np.argmax(Q[state[0], state[1]]) # Greedy action
(exploitation)

            # Get the new state after taking the action
            action = actions[action_idx]
            next_state = (state[0] + action[0], state[1] + action[1])

            # Check if the move is valid
            if not is_valid_move(next_state):
```

```

        next_state = state # Stay in the same position if invalid move

        # Get reward for the new state
        reward = reward_grid[next_state]

        # Q-value update
        best_next_action = np.max(Q[next_state[0], next_state[1]]) # max_a'
Q(s', a')
        Q[state[0], state[1], action_idx] += alpha * (reward + gamma *
best_next_action - Q[state[0], state[1], action_idx])

        # Update state
        state = next_state
        total_reward += reward

        # Check if we reached the goal
        if state == goal:
            break

        # Print progress every 100 episodes
        if episode % 100 == 0:
            print(f"Episode {episode}/{episodes}, Total Reward: {total_reward}")

train_q_learning()

```

#### Step 4: Evaluate the Learned Policy

Once training is complete, we can evaluate the agent's learned policy by simulating its actions in the grid.

```

# Function to evaluate the learned policy
def evaluate_policy():
    state = (0, 0) # Start at the top-left corner
    path = [state] # To track the path taken by the agent

    while state != goal:
        action_idx = np.argmax(Q[state[0], state[1]]) # Choose the best action
(greedy)
        action = actions[action_idx]
        next_state = (state[0] + action[0], state[1] + action[1])

        # Check if the move is valid
        if not is_valid_move(next_state):
            next_state = state # Stay in the same position if invalid move

        state = next_state
        path.append(state)

    return path

# Visualize the path taken by the agent
path = evaluate_policy()
print("Path taken by the agent:", path)

# Create a grid visualization
def visualize_path(path):
    grid = np.zeros((grid_size, grid_size), dtype=int)
    for (y, x) in path:
        grid[y, x] = 1 # Mark the path

```

```
# Print grid with path
for row in grid:
    print(" ".join(["#" if cell == 1 else "." for cell in row]))

visualize_path(path)
```

### Explanation:

1. **Grid Setup:** The grid is represented by a 5x5 matrix. The agent starts at the top-left (0, 0) and the goal is at (4, 4). Obstacles are placed at certain positions, and the agent cannot pass through them.
2. **Q-Learning Algorithm:**
  - o The agent starts at (0, 0) and moves around the grid.
  - o The agent uses **epsilon-greedy** policy: It either explores (random action) or exploits (chooses the best-known action).
  - o For each action, the Q-value is updated using the **Q-learning update rule**.
  - o The agent continues training for a specified number of episodes (1000 in this case).
3. **Evaluation:**
  - o After training, we evaluate the agent by letting it follow the learned policy. The agent chooses actions greedily based on the Q-values.
  - o The path the agent takes from start to goal is displayed.
4. **Result Visualization:** The grid with the path the agent took is visualized, where # represents the path the agent took.

### Conclusion:

This simple example demonstrates how **Q-Learning** can be applied to a decision-making problem (navigating a grid) where the agent learns to maximize cumulative rewards (reaching the goal). By exploring and exploiting the environment, the agent updates its Q-values and improves its policy over time.

## PRACTICAL – 8

### **Aim: Utilizing Transfer Learning to Improve Model Performance on Limited Datasets**

**Transfer learning** is a powerful technique in deep learning where knowledge gained from training a model on a large, diverse dataset is applied to a new, typically smaller dataset. It leverages pre-trained models, which have already learned features that can be reused for similar tasks, thereby improving the performance on the new task with relatively less data.

Transfer learning is particularly useful when the new dataset is limited or lacks sufficient labeled data for training a model from scratch.

### **Transfer Learning Using a Pre-Trained Model**

#### **Step 1: Install Dependencies**

If you haven't installed TensorFlow, use the following command:

```
pip install tensorflow
```

#### **Step 2: Load a Pre-Trained Model**

We will use the **ResNet50** model, which is a widely used architecture pre-trained on the **ImageNet** dataset. We'll load it without the top classification layers so we can adapt it to our own task.

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Load the pre-trained ResNet50 model, without the top layer (classification layer)
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the base model layers to prevent them from being updated during training
base_model.trainable = False
```

#### **Step 3: Add Custom Layers**

After the pre-trained model, we add custom layers that are specific to the new task. These layers will learn from the new dataset.

```
# Create a custom model by adding layers on top of the base ResNet50 model
model = models.Sequential()

# Add the pre-trained ResNet50 model
model.add(base_model)
```

```
# Add custom layers (GlobalAveragePooling2D, Dense layer, and output layer)
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid')) # For binary classification
# (adjust for multi-class)

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

## Step 4: Prepare the Dataset

Since the ResNet50 model expects input images of size **224x224x3**, we'll preprocess the images accordingly. You can use an existing dataset or a custom dataset of images. For the sake of illustration, let's assume you're using a binary classification problem with limited data.

To augment the data (using techniques like rotation, flipping, and scaling), we use **ImageDataGenerator**.

```
# Use ImageDataGenerator for real-time data augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255, # Normalize image pixel values to [0, 1]
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Define validation data generator
validation_datagen = ImageDataGenerator(rescale=1./255)

# Prepare the data from directories (use appropriate paths for your dataset)
train_generator = train_datagen.flow_from_directory(
    'path_to_train_data', # Specify the path to the training data
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary' # Change this to 'categorical' for multi-class
    classification
)

validation_generator = validation_datagen.flow_from_directory(
    'path_to_validation_data', # Specify the path to the validation data
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary' # Change this to 'categorical' for multi-class
    classification
)
```

## Step 5: Fine-Tuning the Model

After training the top layers for a few epochs, you can fine-tune the pre-trained layers by unfreezing some of the layers in the base model.

```

# Unfreeze some layers of the base model for fine-tuning
base_model.trainable = True
# Fine-tune from a certain layer onwards (e.g., unfreeze the last 10 layers)
for layer in base_model.layers[:-10]:
    layer.trainable = False

# Recompile the model after unfreezing the layers
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
               loss='binary_crossentropy', metrics=['accuracy'])

# Fine-tune the model
history = model.fit(
    train_generator,
    epochs=10, # Number of epochs to fine-tune the model
    validation_data=validation_generator
)

```

## Step 6: Evaluate the Model

After training and fine-tuning, evaluate the model on the validation data to see how well it performs.

```

# Evaluate the model
test_loss, test_acc = model.evaluate(validation_generator)
print(f'Test Accuracy: {test_acc}')

```

## Step 7: Save the Model

Once the model is trained, you can save it for later use or deployment.

```

# Save the model
model.save('transfer_learning_model.h5')

```

## Benefits of Transfer Learning

- **Improved Performance on Small Datasets:** Transfer learning allows us to achieve better results on limited data by leveraging pre-trained models that already have learned rich features.
- **Faster Training:** Since the base model has already been trained, we can freeze most of its layers, reducing the training time and computational cost.
- **Lower Risk of Overfitting:** Fine-tuning a pre-trained model helps prevent overfitting, as the model starts with weights that already capture general features from a large dataset (e.g., ImageNet).

## Conclusion

Using transfer learning, we can dramatically improve model performance, especially on tasks with limited datasets, by leveraging the knowledge from large-scale pre-trained models. Fine-tuning these models for specific tasks helps in adapting them to new challenges, such as classifying images in a small dataset.

PRACTICAL – 9**Aim: Building a Deep Learning Model for Time Series Forecasting or Anomaly Detection**

Time series forecasting and anomaly detection are crucial tasks in various fields like finance, healthcare, and industry. Deep learning models, such as **Recurrent Neural Networks (RNNs)**, **Long Short-Term Memory (LSTM)** networks, and **Gated Recurrent Units (GRUs)**, are particularly suited for time series tasks because they are designed to handle sequential data.

1. **Build a deep learning model for time series forecasting using LSTM** (a type of RNN).
2. **Build a deep learning model for anomaly detection** in time series using an **autoencoder**.

Both models will use **TensorFlow** and **Keras**.

**Part 1: Time Series Forecasting with LSTM**

In time series forecasting, the goal is to predict future values of a time series based on past data.

**Step 1: Install Dependencies**

Install the necessary Python libraries if you haven't already:

```
pip install tensorflow numpy pandas matplotlib scikit-learn
```

**Step 2: Load and Preprocess Data**

We'll use the **Airline Passengers Dataset** as an example for time series forecasting. It contains monthly total passenger counts from 1949 to 1960. We will predict future passenger counts based on past data.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

# Load the dataset (you can replace this with any other time series dataset)
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/airline-passengers.csv"
data = pd.read_csv(url, usecols=[1], engine='python', header=0)

# Visualize the data
plt.plot(data)
plt.title('Monthly Airline Passengers')
plt.xlabel('Month')
plt.ylabel('Passengers')
plt.show()

# Normalize the data (scaling to [0, 1] range for LSTM)
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled = scaler.fit_transform(data)

# Function to create the dataset in X (input) and y (output) for LSTM
```

```

def create_dataset(data, time_step=1):
    X, y = [], []
    for i in range(len(data)-time_step-1):
        X.append(data[i:(i+time_step), 0])
        y.append(data[i + time_step, 0])
    return np.array(X), np.array(y)

# Prepare the data for LSTM
time_step = 12 # Use 12 previous months to predict the next month
X, y = create_dataset(data_scaled, time_step)

# Reshape X for LSTM input: [samples, time steps, features]
X = X.reshape(X.shape[0], X.shape[1], 1)

```

### Step 3: Build the LSTM Model

Now we define the **LSTM model** architecture:

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=False, input_shape=(X.shape[1], 1)))
model.add(Dropout(0.2)) # Dropout for regularization
model.add(Dense(units=1)) # Output layer (single value for forecasting)

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

```

### Step 4: Train the Model

We will train the model on the time series data:

```
# Train the model
model.fit(X, y, epochs=20, batch_size=32)
```

### Step 5: Make Predictions

Now that the model is trained, we can use it to make predictions:

```

# Predict the next 12 months
test_input = data_scaled[-time_step:] # Last 12 months of data for forecasting
test_input = test_input.reshape(1, -1, 1) # Reshape for LSTM input

# Predict
predicted = model.predict(test_input)
predicted = scaler.inverse_transform(predicted) # Inverse scaling

print(f'Predicted passengers for the next month: {predicted[0][0]}')

```

### Step 6: Visualize the Predictions

```

# Plot the results
train_data = data[:len(data) - 12]
plt.plot(train_data, label='Training Data')
plt.plot(range(len(train_data), len(train_data) + 12), predicted, label='Forecast',
color='red')

```

```
plt.legend()
plt.title('Time Series Forecasting')
plt.xlabel('Month')
plt.ylabel('Passengers')
plt.show()
```

## Anomaly Detection with Autoencoders

Anomaly detection in time series is the task of identifying unusual patterns or outliers. An **autoencoder** is an unsupervised deep learning model used for anomaly detection. The autoencoder learns to compress (encode) the input into a latent representation and then reconstruct (decode) it. If the reconstruction error is high, the data point is likely an anomaly.

### Step 1: Build the Autoencoder Model

We will build an autoencoder for anomaly detection using time series data.

```
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model

# Define the autoencoder model
input_dim = X.shape[1]
encoding_dim = 14 # Number of dimensions for the encoded representation

input_layer = Input(shape=(input_dim,))
encoded = Dense(encoding_dim, activation='relu')(input_layer)
decoded = Dense(input_dim, activation='sigmoid')(encoded)

# Create the autoencoder model
autoencoder = Model(inputs=input_layer, outputs=decoded)

# Compile the model
autoencoder.compile(optimizer='adam', loss='mean_squared_error')

# Train the autoencoder on the normal (non-anomalous) data
autoencoder.fit(X, X, epochs=50, batch_size=32)
```

### Step 2: Detect Anomalies

We can now detect anomalies by checking the reconstruction error. If the error exceeds a certain threshold, the data point is considered anomalous.

```
# Make predictions using the autoencoder
reconstructed = autoencoder.predict(X)

# Calculate reconstruction error (Mean Squared Error)
reconstruction_error = np.mean(np.square(X - reconstructed), axis=1)

# Set a threshold for anomaly detection (this can be tuned)
threshold = np.percentile(reconstruction_error, 95)

# Identify anomalies (where reconstruction error is higher than threshold)
anomalies = reconstruction_error > threshold
```

```
# Visualize anomalies
plt.plot(data)
plt.scatter(np.where(anomalies)[0], data[anomalies], color='red', label='Anomalies')
plt.title('Anomaly Detection in Time Series')
plt.xlabel('Time')
plt.ylabel('Value')
plt.legend()
plt.show()
```

### Step 3: Evaluate the Results

To evaluate the anomaly detection, you could:

- Use labeled datasets with known anomalies.
- Analyze the identified anomalies and compare them with known outliers.

### Conclusion

- **LSTM for Forecasting:** This model is suitable for forecasting future values in time series data. It captures temporal dependencies and is effective when historical data patterns are crucial to predict future outcomes.
- **Autoencoders for Anomaly Detection:** Autoencoders are great for detecting anomalies in time series data because they learn the typical patterns and can highlight unusual or out-of-norm events by examining reconstruction errors.

## PRACTICAL – 10

### **Aim: Implementing a Machine Learning Pipeline for Automated Feature Engineering and Model Selection**

A **Machine Learning Pipeline** automates various steps in the process of building and deploying machine learning models. The pipeline typically includes:

1. Data preprocessing and feature engineering
2. Model selection and evaluation
3. Hyperparameter tuning
4. Model training and deployment

### **Automated Machine Learning Pipeline**

We'll demonstrate the pipeline using **scikit-learn** and the **TPOT** library for automated machine learning. TPOT helps automatically select the best features, models, and hyperparameters for your task.

#### **Step 1: Install Dependencies**

Install the necessary Python libraries:

```
pip install scikit-learn tpot optuna pandas numpy matplotlib
```

#### **Step 2: Load and Preprocess Data**

For the sake of illustration, we'll use the **Iris dataset** from **scikit-learn** (you can replace this with your own dataset):

```
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Feature scaling (important for algorithms like SVM and neural networks)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

#### **Step 3: Automated Feature Engineering (Optional)**

While basic preprocessing like scaling is covered, more advanced feature engineering can be done using libraries like **Feature-engine** or **TPOT**. For this basic example, we'll focus on preprocessing using **scikit-learn**.

If you had more complex data (with categorical variables, missing values, etc.), automated feature engineering might involve:

- Encoding categorical variables (e.g., One-Hot Encoding, Label Encoding)
- Imputation of missing values
- Feature extraction techniques (e.g., PCA, polynomial features, etc.)

#### **Step 4: Use TPOT for Automated Model Selection and Hyperparameter Tuning**

**TPOT** uses genetic algorithms to automatically search for the best model and preprocessing pipeline. Here, we'll use TPOT to automate the selection of machine learning models and hyperparameters.

```
from tpot import TPOTClassifier

# Initialize the TPOTClassifier
tpot = TPOTClassifier(generations=5, population_size=20, random_state=42, cv=5,
verbosity=2)

# Train the model (TPOT automatically handles preprocessing and model selection)
tpot.fit(X_train_scaled, y_train)

# Evaluate the model
accuracy = tpot.score(X_test_scaled, y_test)
print(f"Test accuracy: {accuracy:.4f}")

# Export the best pipeline
tpot.export('best_model_pipeline.py')
```

#### **Step 5: Automated Hyperparameter Tuning with Optuna (Optional)**

In addition to using TPOT, we can also use **Optuna** for hyperparameter optimization. Optuna allows you to define a search space and automatically search for the best hyperparameters using techniques like **Tree-structured Parzen Estimator (TPE)**.

Here is an example using **Optuna** to optimize hyperparameters for a **Random Forest Classifier**.

```
import optuna
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Define the objective function for optimization
def objective(trial):
    # Hyperparameter space
    n_estimators = trial.suggest_int('n_estimators', 50, 200)
    max_depth = trial.suggest_int('max_depth', 5, 20)
    min_samples_split = trial.suggest_int('min_samples_split', 2, 10)

    # Train the model with the selected hyperparameters
    model = RandomForestClassifier(n_estimators=n_estimators, max_depth=max_depth,
    min_samples_split=min_samples_split, random_state=42)
    model.fit(X_train_scaled, y_train)

    # Evaluate the model
    accuracy = model.score(X_test_scaled, y_test)
    return accuracy
```

```

# Predict and calculate accuracy
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
return accuracy

# Create an Optuna study and optimize
study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=20)

# Print the best hyperparameters found
print("Best hyperparameters:", study.best_params)

# Train the model with the best hyperparameters
best_params = study.best_params
best_model = RandomForestClassifier(**best_params, random_state=42)
best_model.fit(X_train_scaled, y_train)

# Evaluate the model
best_accuracy = best_model.score(X_test_scaled, y_test)
print(f"Best Model Test Accuracy: {best_accuracy:.4f}")

```

### Step 6: Evaluate the Best Model

After training and tuning, you can evaluate the model's performance on the test set. Both **TPOT** and **Optuna** return models with optimized parameters, which you can then use to predict on new data.

```

# Evaluate the best model from TPOT or Optuna
y_pred = best_model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(f"Test Accuracy of Final Model: {accuracy:.4f}")

```

### Step 7: Model Deployment (Optional)

Once you have the best model, you can save it for future use or deployment using `joblib` or `pickle`.

```

import joblib

# Save the trained model
joblib.dump(best_model, 'best_model.pkl')

# Save the scaler for future use
joblib.dump(scaler, 'scaler.pkl')

```

This allows you to reload the model and use it for predictions on new, unseen data.



NURTURING POTENTIAL

SAKET GYANPEETH'S  
**SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE**  
(Permanently Affiliated to University of Mumbai)

NAAC Accredited

Saket Vidyanagri Marg, Chinchpada Road, Katemanivali,  
Kalyan (East) -421306(Mah)

**Department of Information Technology**

This is to certify that

Mr./Ms. \_\_\_\_\_ Seat No. \_\_\_\_\_

of

**M.Sc. Information Technology**

**Part II NEP 2020 Semester III**

has satisfactorily carried out the required practical in the subject  
of \_\_\_\_\_

For the Academic year 2024 – 2025

---

**Practical In-Charge**

---

**Head of the Department**

---

**External Examiner**

**College Seal**



NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE

KALYAN (EAST)

ACADEMIC YEAR 2024-25

**M.Sc. Information Technology**

**Part II NEP 2020 Semester III**

**SUBMITTED BY**

---

**AS PRESCRIBED BY**

**UNIVERSITY OF MUMBAI**



**MUMBAI UNIVERSITY**

## INDEX

<u>Sr. No.</u>	<u>Practical Aim</u>	<u>Signature</u>
1.	<p><b>Data Pre-processing and Exploration</b></p> <ul style="list-style-type: none"> <li>a. Load a CSV dataset. Handle missing values, inconsistent formatting, and outliers.</li> <li>b. Load a dataset, calculate descriptive summary statistics, create visualizations using different graphs, and identify potential features and target variables Explore Univariate and Bivariate graphs (Matplotlib) and Seaborn for visualization.</li> <li>c. Create or Explore datasets to use all pre-processing routines like label encoding, scaling, and binarization.</li> </ul>	
2.	<p><b>Testing Hypothesis</b></p> <ul style="list-style-type: none"> <li>a. Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a CSV file and generate the final specific hypothesis. (Create your dataset)</li> </ul>	
3.	<p><b>Linear Models</b></p> <ul style="list-style-type: none"> <li>a. <b>Simple Linear Regression</b> Fit a linear regression model on a dataset. Interpret coefficients, make predictions, and evaluate performance using metrics like R-squared and MSE.</li> <li>b. <b>Multiple Linear Regression</b> Extend linear regression to multiple features. Handle feature selection and potential multicollinearity.</li> <li>c. <b>Regularized Linear Models</b> (Ridge, Lasso, Elastic Net) Implement regression variants like LASSO and Ridge on any generated dataset</li> </ul>	
4.	<p><b>Discriminative Models</b></p> <ul style="list-style-type: none"> <li>a. <b>Logistic Regression</b> Perform binary classification using logistic regression. Calculate accuracy, precision, recall, and understand the ROC curve.</li> <li>b. Implement and demonstrate k-nearest Neighbor algorithm. Read the training data from a .CSV file and build the model to classify a test sample. Print both correct and wrong predictions.</li> </ul>	

	<ul style="list-style-type: none"> <li>c. Build a decision tree classifier or regressor. Control hyper parameters like tree depth to avoid overfitting. Visualize the tree.</li> <li>d. Implement a Support Vector Machine for any relevant dataset.</li> <li>e. Train a random forest ensemble. Experiment with the number of trees and feature sampling. Compare performance to a single decision tree.</li> <li>f. Implement a gradient boosting machine (e.g., XGBoost). Tune hyper parameters and explore feature importance.</li> </ul>	
5.	<p><b>Generative Models</b></p> <ul style="list-style-type: none"> <li>a. Implement and demonstrate the working of a Naive Bayesian classifier using a sample data set. Build the model to classify a test sample.</li> <li>b. Implement Hidden Markov Models using hmmlearn</li> </ul>	
6.	<p><b>Probabilistic Models</b></p> <ul style="list-style-type: none"> <li>a. Implement Bayesian Linear Regression to explore prior and posterior distribution.</li> <li>b. Implement Gaussian Mixture Models for density estimation and unsupervised clustering</li> </ul>	
7.	<p><b>Model Evaluation and Hyper parameter Tuning</b></p> <ul style="list-style-type: none"> <li>a. Implement cross-validation techniques (k-fold, stratified, etc.) for robust model evaluation.</li> <li>b. Systematically explore combinations of hyper parameters to optimize model performance.(use grid and randomized search)</li> </ul>	
8.	<p><b>Bayesian Learning</b></p> <ul style="list-style-type: none"> <li>a. Implement Bayesian Learning using inferences</li> </ul>	

## **Practical 1: Data Pre-processing and Exploration**

**1a. Load a CSV dataset. Handle missing values, inconsistent formatting, and outliers.**

**Code :**

### **1. Import Libraries**

**# Import necessary libraries**

```
import pandas as pd import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt
```

### **2. Load the Dataset**

**# Load the Titanic dataset from a URL**

```
url="https://raw.githubusercontent.com/datasets/master/titanic.csv" data =  
pd.read_csv(url)
```

**# Display the first few rows**

```
print(data.head())
```

### **3. Handle Missing Values**

**# Check for missing values**

```
print("Missing values in each column:")
```

```
print(data.isnull().sum())
```

**# Fill missing values in 'Age' with the mean**

```
data['Age'].fillna(data['Age'].mean(), inplace=True)
```

**# Fill missing values in 'Embarked' with the most common value**

```
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
```

**# Drop rows where 'Cabin' is missing (too many NaNs)**

```
data.drop(columns=['Cabin'], inplace=True)
```

**# Verify missing values are handled**

```
print("\nAfter handling missing values:")
print(data.isnull().sum())
```

#### 4. Fix Inconsistent Formatting

```
# Fix inconsistent formatting in the 'Sex' column
```

```
data['Sex'] = data['Sex'].str.lower().str.strip()
```

```
# Verify unique values
```

```
print("\nUnique values in 'Sex' column after formatting:")
```

```
print(data['Sex'].unique())
```

```
5. Detect and Handle Outliers # Boxplot for the 'Fare' column sns.boxplot(data['Fare'],
color='skyblue') plt.title('Boxplot of Fare') plt.show()
```

```
# Detect outliers using the IQR method
```

```
Q1 = data['Fare'].quantile(0.25)
Q3 = data['Fare'].quantile(0.75)
IQR = Q3 - Q1
lower_bound =
Q1 - 1.5 * IQR
upper_bound =
Q3 + 1.5 * IQR
```

```
# Capping outliers
```

```
data['Fare'] = np.where(data['Fare'] > upper_bound, upper_bound, np.where(data['Fare'] <
lower_bound, lower_bound, data['Fare']))
```

```
# Verify with an updated boxplot
```

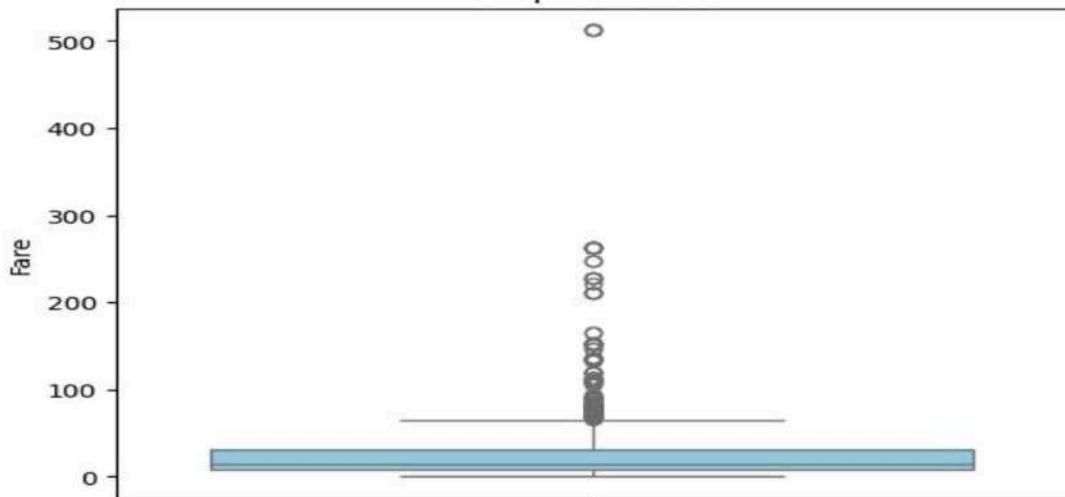
```
sns.boxplot(data['Fare'], color='lightgreen')
plt.title('Boxplot of Fare (After Handling Outliers)')
plt.show()
```

```
6. Save the Cleaned Dataset # Save the cleaned dataset
```

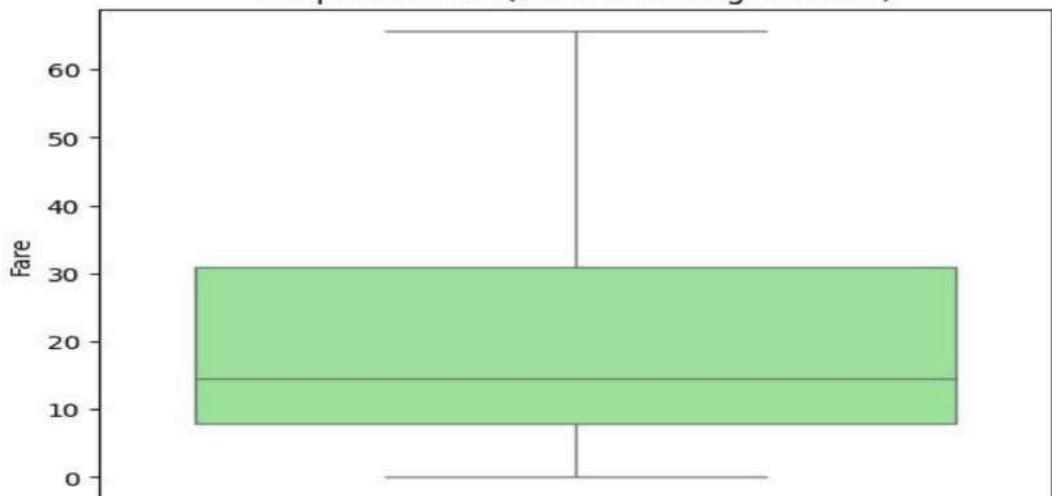
```
data.to_csv('cleaned_titanic.csv', index=False)
print("\nCleaned dataset saved as 'cleaned_titanic.csv'") .
```

**Output :**

**Boxplot of Fare**



**Boxplot of Fare (After Handling Outliers)**



**1b. Load a dataset, calculate descriptive summary statistics, create visualizations using different graphs, and identify potential features and target variables Note:**

**Explore Univariate and Bivariate graphs (Matplotlib) and Seaborn for visualization**

**Code :**

**1. Import Necessary Libraries # Import required libraries**

```
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns
```

**2. Load the Dataset**

**# Load the dataset from the URL**

```
url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv"  
data = pd.read_csv(url)  
  
# Display the first few rows  
  
print("First 5 rows of the dataset:")  
print(data.head())
```

**3. Calculate Descriptive Summary Statistics # Dataset information**

```
print("\nDataset Info:")  
print(data.info())  
  
# Summary statistics for numerical columns  
  
print("\nDescriptive Statistics for Numerical Columns:")  
print(data.describe())  
  
# Check unique values for categorical columns  
  
print("\nUnique values in 'species' column:")  
print(data['species'].value_counts())
```

**4. Univariate Analysis**

**# Histograms for numerical columns**

```
data.hist(figsize=(10,8), color='skyblue', edgecolor='black')
plt.suptitle("Histograms of Numerical Features")
plt.show()

# Bar plot for 'species' column
sns.countplot(x='species', data=data, palette='pastel')
plt.title("Count of Each Species") plt.show()
```

## 5. Bivariate Analysis

### # Scatter plot for two features

```
plt.figure(figsize=(8, 6))

plt.scatter(data['sepal_length'], data['sepal_width'], alpha=0.7, c='blue')
plt.title("Sepal Length vs Sepal Width")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.show()
```

### # Pairplot to visualize relationships between features

```
sns.pairplot(data, hue='species', palette='husl', diag_kind='kde')
plt.suptitle("Pairplot of Features by Species", y=1.02)
plt.show()
```

```
# Boxplot for petal_length across species sns.boxplot(x='species',
y='petal_length', data=data, palette='Set3')

plt.title("Boxplot of Petal Length by Species")
plt.show()
```

## 6. Identify Potential Features and Target Variables

### # Separate features and target

```
features = data.drop(columns=['species'])
```

### # Drop the target

```
column_target = data['species']
```

### # Target variable

```
print("\nFeatures:")
```

```
print(features.head())
```

```
print("\nTarget:")
```

```
print(target.head())
```

```
# Visualize target distribution
sns.countplot(x=target, palette='viridis')
plt.title("Target Variable Distribution")

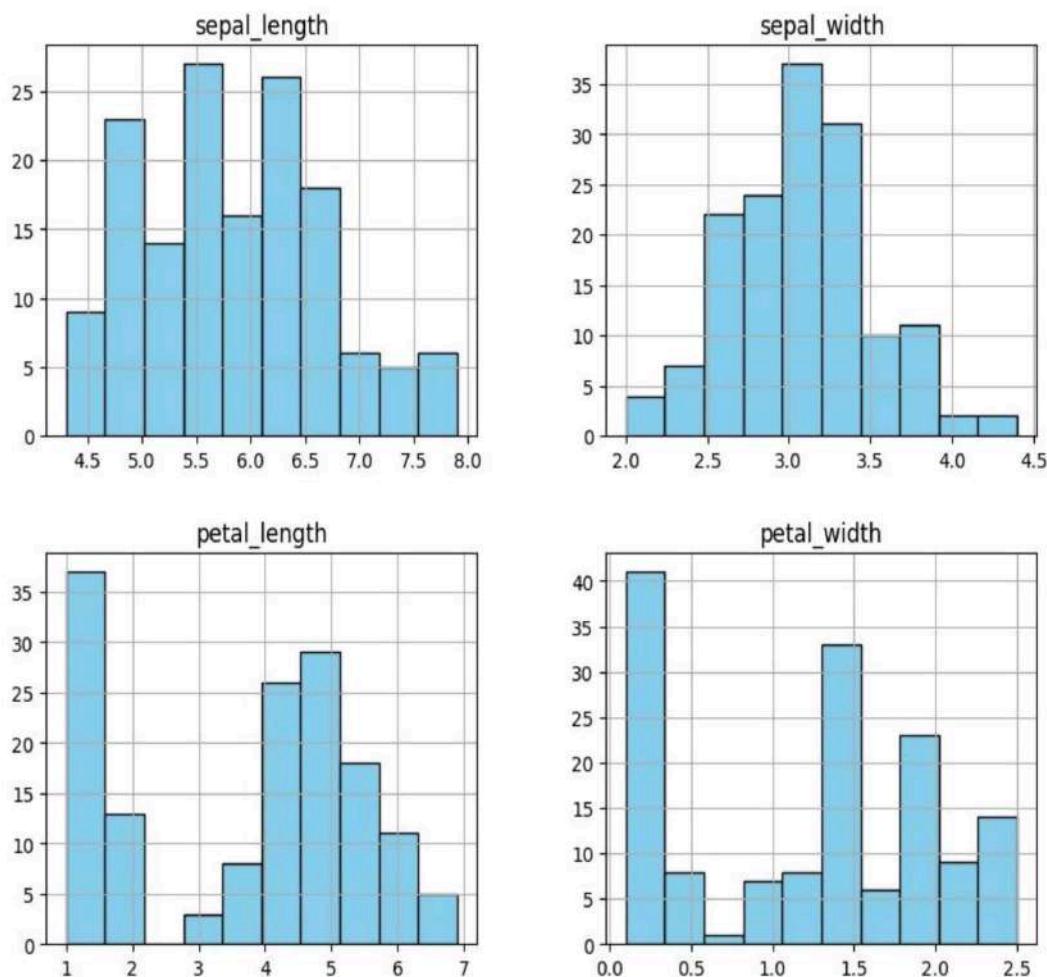
plt.show()
```

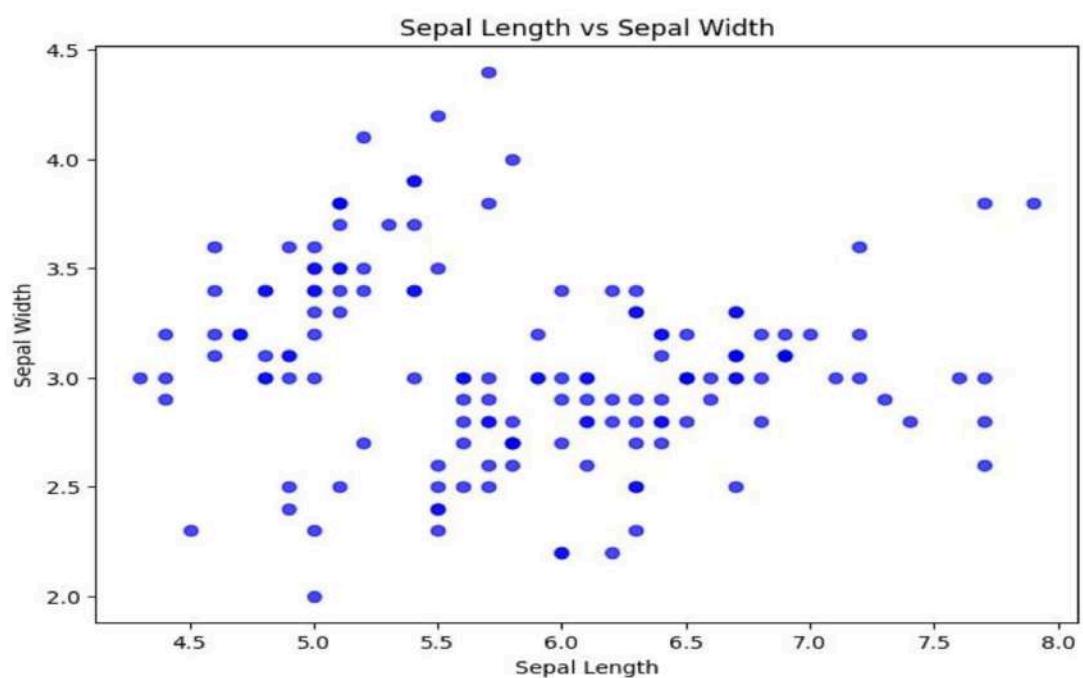
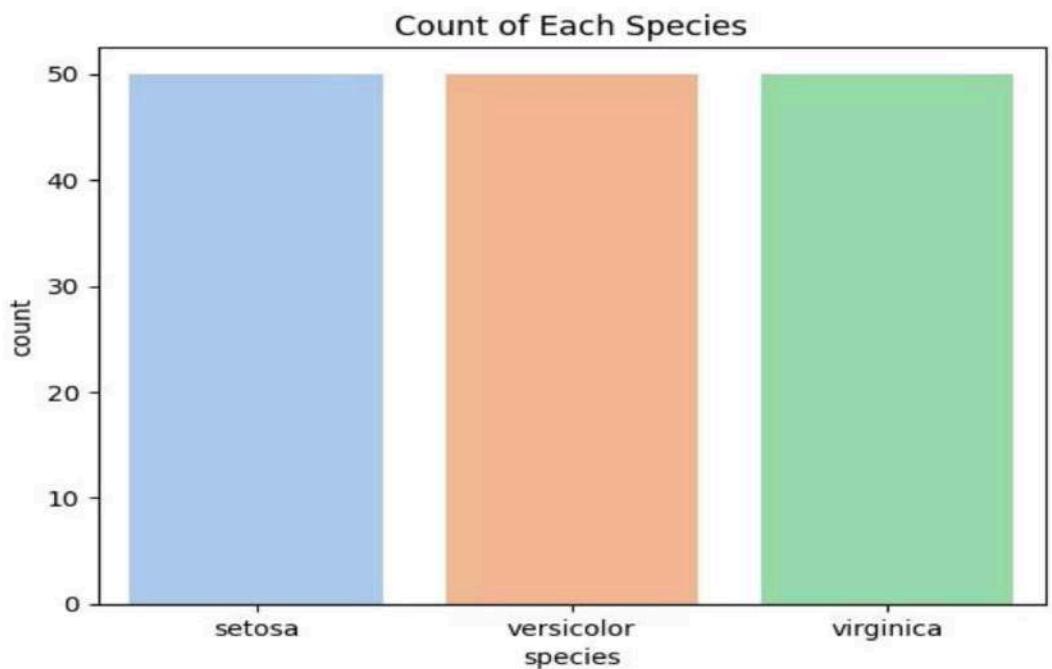
## 7. Save the Cleaned and Processed Dataset

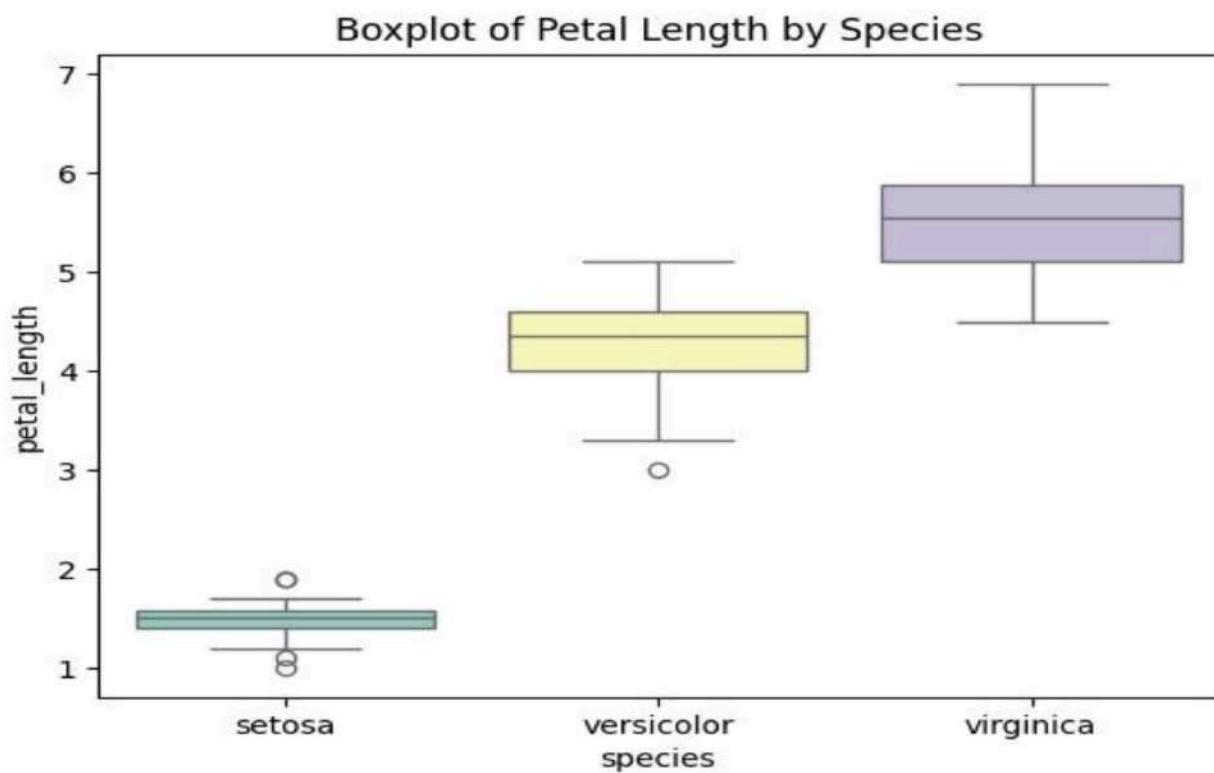
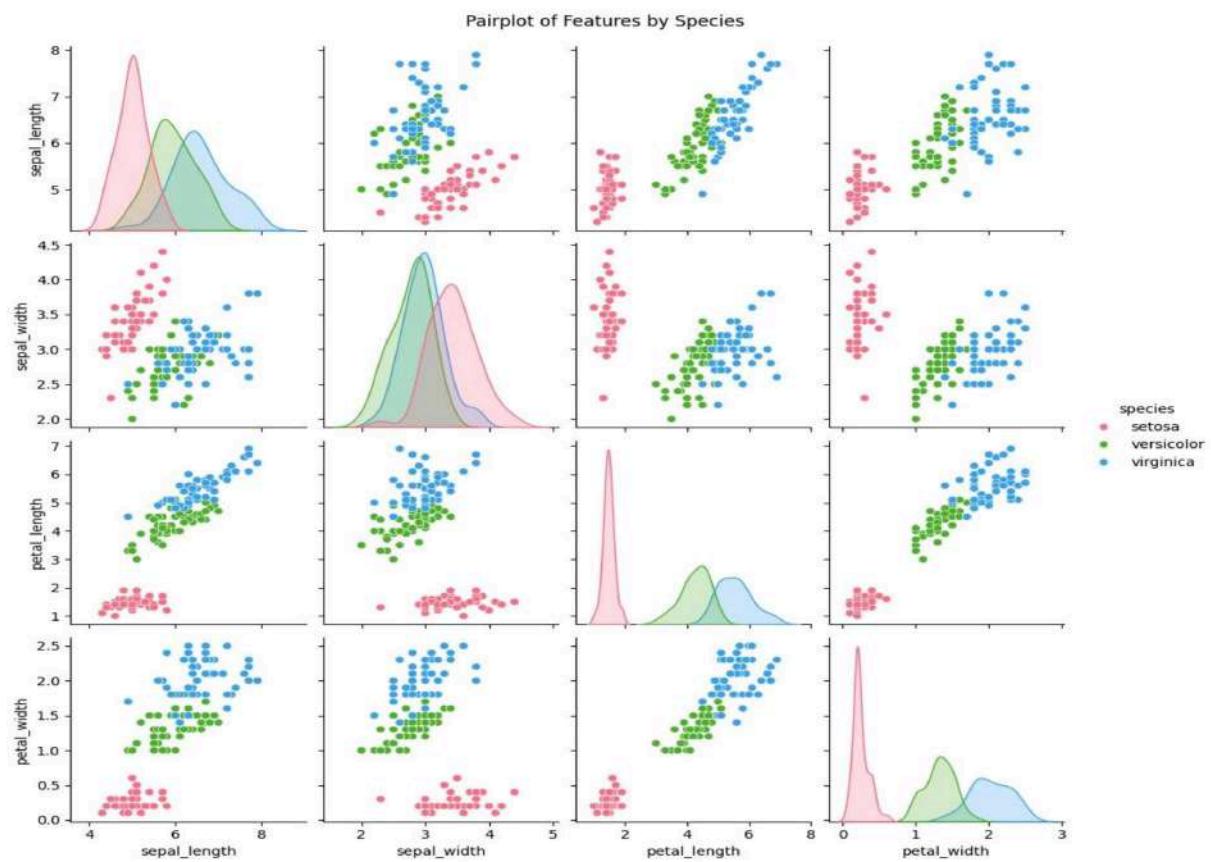
```
# Save the dataset
data.to_csv('processed_iris.csv', index=False)
print("\nProcessed dataset saved as 'processed_iris.csv'")
```

## Output :

Histograms of Numerical Features







**1c. Create or Explore datasets to use all pre-processing routines like label encoding, scaling, and binarization.**

**Code :**

**1. Import Necessary Libraries # Import required libraries**

```
import pandas as pd  
import numpy as np from sklearn.preprocessing  
import LabelEncoder, MinMaxScaler, StandardScaler, Binarizer
```

**2. Create or Load a Dataset # Create a sample dataset**

```
data = pd.DataFrame({  
    'Category': ['A', 'B', 'C', 'A', 'B', 'C'],  
    # Categorical variable  
    'Age': [23, 45, 31, 22, 35, 30],  
    # Numerical variable  
    'Income': [50000, 60000, 70000, 80000, 90000, 100000],  
    # Numerical variable 'Has_Car':  
    ['Yes', 'No', 'Yes', 'No', 'Yes', 'No']  
    # Binary categorical variable })
```

**# Display the dataset**

```
print("Sample Dataset:")  
print(data)
```

**3. Apply Pre-Processing Routines**

**# Label Encoding for 'Category' column**

```
label_encoder = LabelEncoder()  
data['Category_Encoded'] =  
label_encoder.fit_transform(data['Category'])  
# Label Encoding for binary column 'Has_Car'
```

```

data['Has_Car_Encoded'] =
label_encoder.fit_transform(data['Has_Car']) print("\nAfter Label
Encoding:")
print(data)

# Min-Max Scaling for 'Income'
min_max_scaler = MinMaxScaler()
data['Income_MinMax'] = min_max_scaler.fit_transform(data[['Income']])

# Standard Scaling for 'Age'
standard_scaler = StandardScaler()
data['Age_Standardized'] =
standard_scaler.fit_transform(data[['Age']]) print("\nAfter Scaling:")
print(data)

# Binarization for 'Income' with a threshold of 75,000
binarizer = Binarizer(threshold=75000)
data['Income_Binary'] =
binarizer.fit_transform(data[['Income']]) print("\nAfter
Binarization:")
print(data)

```

#### **4. Save the Processed Dataset**

```

# Save the processed dataset
data.to_csv('processed_data.csv', index=False)
print("\nProcessed dataset saved as
'processed_data.csv'")

```

**Output :**

Sample Dataset:					
	Category	Age	Income	Has_Car	
0	A	23	50000	Yes	
1	B	45	60000	No	
2	C	31	70000	Yes	
3	A	22	80000	No	
4	B	35	90000	Yes	
5	C	30	100000	No	



After Label Encoding:

	Category	Age	Income	Has_Car	Category_Encoded	Has_Car_Encoded
0	A	23	50000	Yes	0	1
1	B	45	60000	No	1	0
2	C	31	70000	Yes	2	1
3	A	22	80000	No	0	0
4	B	35	90000	Yes	1	1
5	C	30	100000	No	2	0



After Scaling:

	Category	Age	Income	Has_Car	Category_Encoded	Has_Car_Encoded	\
0	A	23	50000	Yes	0	1	
1	B	45	60000	No	1	0	
2	C	31	70000	Yes	2	1	
3	A	22	80000	No	0	0	
4	B	35	90000	Yes	1	1	
5	C	30	100000	No	2	0	

	Income_MinMax	Age_Standardized	
0	0.0	-1.035676	
1	0.2	1.812434	
2	0.4	0.000000	
3	0.6	-1.165136	
4	0.8	0.517838	
5	1.0	-0.129460	



After Binarization:

	Category	Age	Income	Has_Car	Category_Encoded	Has_Car_Encoded	\
0	A	23	50000	Yes	0	1	
1	B	45	60000	No	1	0	
2	C	31	70000	Yes	2	1	
3	A	22	80000	No	0	0	
4	B	35	90000	Yes	1	1	
5	C	30	100000	No	2	0	

	Income_MinMax	Age_Standardized	Income_Binary
0	0.0	-1.035676	0
1	0.2	1.812434	0
2	0.4	0.000000	0
3	0.6	-1.165136	1
4	0.8	0.517838	1
5	1.0	-0.129460	1

## Practical 2 : Testing Hypothesis

**Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a. CSV file and generate the final specific hypothesis. (Create your dataset)**

### **CODE :**

```
import pandas as pd

# Step 1: Create the Dataset and Load It

data = {'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny',
'Sunny', 'Rainy'],
'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Mild', 'Cool', 'Mild'],
'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal',
'Normal'],
'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Weak', 'Weak'],
'PlayTennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes']
}

}
```

### **# Load dataset into a pandas DataFrame**

```
df = pd.DataFrame(data)

# Step 2: Implementing the FIND-S Algorithm

def find_s_algorithm(data):

    # Get the positive examples (PlayTennis = 'Yes')
    positive_examples = data[data['PlayTennis'] == 'Yes']

    # Initialize hypothesis with the first positive example (most specific)
    hypothesis = positive_examples.iloc[0].drop('PlayTennis')

    # Loop through the rest of the positive examples and generalize the
    # hypothesis

    for index, row in positive_examples.iterrows():

        for feature in hypothesis.index:

            if hypothesis[feature] != row[feature]:
                hypothesis[feature] = '?'
```

```
    return hypothesis
```

### Step 3: Apply FIND-S to the dataset

```
hypothesis = find_s_algorithm(df)

# Display the final specific hypothesis

print("The most specific hypothesis is:")

print(hypothesis)
```

### Output :

```
→ Dataset:
   Sky Temperature Humidity      Wind Water Forecast Condition
0  Sunny          Warm  Normal  Strong  Warm    Same     Yes
1  Sunny          Cold   High   Strong  Warm    Same     No
2  Rainy          Warm   High   Weak   Cool   Change   No
3  Sunny          Warm  Normal  Strong  Warm    Same     Yes
4  Rainy          Cold  Normal   Weak   Cool   Change   No
```

```
→
Loaded Dataset:
   Sky Temperature Humidity      Wind Water Forecast Condition
0  Sunny          Warm  Normal  Strong  Warm    Same     Yes
1  Sunny          Cold   High   Strong  Warm    Same     No
2  Rainy          Warm   High   Weak   Cool   Change   No
3  Sunny          Warm  Normal  Strong  Warm    Same     Yes
4  Rainy          Cold  Normal   Weak   Cool   Change   No
```

```
→
Final Specific Hypothesis:
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
```

## Practical 3 : Linear Models

### **3a. Simple Linear Regression**

Fit a linear regression model on a dataset. Interpret coefficients, make predictions, and evaluate performance using metrics like R-squared and MSE

**Code :**

#### **Step 1: Import Libraries # Import required libraries**

```
import pandas as pd  
  
import numpy as np  
  
import matplotlib.pyplot as plt  
  
from sklearn.model_selection import train_test_split  
  
from sklearn.linear_model import LinearRegression  
  
from sklearn.metrics import mean_squared_error, r2_score
```

#### **Step 2: Create a Dataset and Save as CSV**

##### **# Create a sample dataset**

```
data = {  
  
    'House_Size': [750, 800, 850, 900, 1000, 1100, 1200, 1300, 1400, 1500],  
  
    'Price': [150000, 160000, 165000, 170000, 180000, 190000, 200000, 210000, 220000,  
    230000]  
}
```

##### **# Convert the dataset into a DataFrame**

```
df = pd.DataFrame(data)
```

##### **# Save to CSV file**

```
df.to_csv('house_prices.csv',  
index=False)
```

##### **# Display the dataset**

```
print("Dataset:")  
print(df)
```

#### **Step 3: Load the Dataset**

```
# Load the dataset  
dataset = pd.read_csv('house_prices.csv')  
# Display the first few  
rows      print("\nLoaded  
Dataset:")  
print(dataset.head())
```

#### **Step 4: Split the Dataset into Training and Test Sets**

```
# Features and target variable  
X = dataset[['House_Size']] # Feature: House size  
y = dataset['Price']      # Target: Price  
  
# Split data into training and testing sets (80% train, 20% test)  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  
print("\nTraining and Testing Data Sizes:")  
print("Training Data Size:", X_train.shape[0])  
print("Testing Data Size:", X_test.shape[0])
```

#### **Step 5: Fit a Linear Regression Model**

```
# Initialize and fit the linear regression model  
model = LinearRegression()  
model.fit(X_train, y_train)  
  
# Display the coefficients  
print("\nModel Coefficients:")  
print("Slope (m):", model.coef_[0])  
print("Intercept (b):", model.intercept_)
```

#### **Step 6: Make Predictions**

```
# Predict on the test set  
y_pred = model.predict(X_test)  
# Display predictions  
print("\nPredictions on Test Data:")  
print("Actual Prices:", y_test.values)  
print("Predicted Prices:", y_pred)
```

#### **Step 7: Evaluate the Model**

```
# Calculate evaluation metrics  
mse = mean_squared_error(y_test, y_pred) r2 = r2_score(y_test, y_pred)
```

## # Display metrics

```
print("\nModel Performance Metrics:")
print("Mean Squared Error (MSE):", mse)
print("R-squared (R2):", r2)
```

## Step 8: Visualize the Results # Scatter plot of the training data

```
plt.scatter(X_train, y_train, color='blue', label='Training Data')
```

## # Plot the regression line

```
plt.plot(X_train, model.predict(X_train), color='red', label='Regression Line')
# Scatter plot of the test data
```

```
plt.scatter(X_test, y_test, color='green', label='Test Data')
```

```
plt.title("Simple Linear Regression")
```

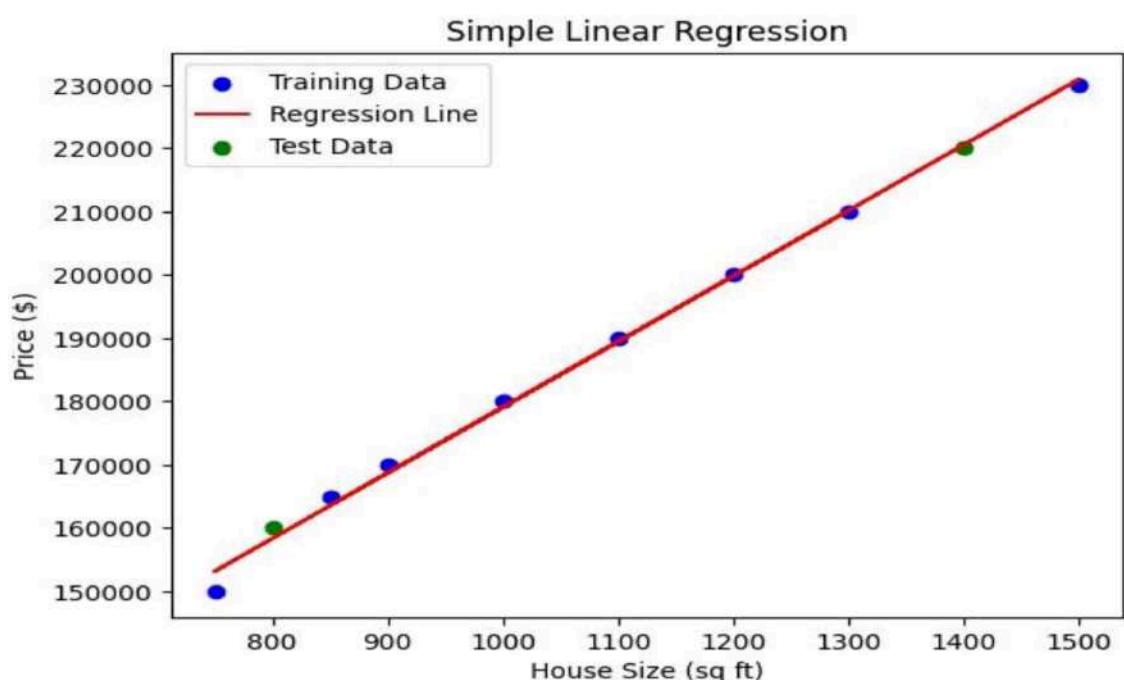
```
plt.xlabel("House Size (sq ft)")
```

```
plt.ylabel("Price ($)")
```

```
plt.legend()
```

```
plt.show()
```

## Output :



### **3b. Multiple Linear Regression :**

Extend linear regression to multiple feature. Handle feature selection and potential multicollinearity

#### **Code :**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import LabelEncoder

# Import LabelEncoder
from sklearn.impute import SimpleImputer

# Load dataset
from google.colab import files
uploaded = files.upload() # Upload your CSV file

# Read the CSV file
data = pd.read_csv(list(uploaded.keys())[0])

# Display the first few rows
print(data.head())

# Check for null values and basic statistics
print(data.info())
print(data.describe())

# Define a function to calculate VIF
def calculate_vif(df):

# Select only numeric features for VIF calculation
numeric_df = df.select_dtypes(include=np.number)

# Drop rows with infinite or missing values
```

```

numeric_df = numeric_df.replace([np.inf, -np.inf], np.nan).dropna()
vif_data = pd.DataFrame()
vif_data["feature"] = numeric_df.columns
vif_data["VIF"] = [variance_inflation_factor(numeric_df.values, i)
for i in range(numeric_df.shape[1])]

# Selecting features and target variable
X = data.drop("Survived", axis=1)
# Changed 'y' to 'Survived' y = data["Survived"]

# Handle categorical features (e.g., using Label Encoding)
for col in X.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])

# Impute missing values using the mean (you can choose other strategies)
imputer = SimpleImputer(strategy='mean')
# Create an imputer instance
X = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)
# Impute and update X

# Calculate VIF for initial features
print("VIF before handling multicollinearity:")
print(calculate_vif(X)) # Call the modified function

# Drop features based on VIF analysis (example: drop 'X1' if VIF is high)
# Check if the column exists before dropping
if 'X1' in X.columns:
    X = X.drop("X1", axis=1) # Replace 'X1' with the actual high VIF feature name
else:
    print("Column 'X1' not found in the DataFrame.")

# Recalculate VIF
print("VIF after handling multicollinearity:")
print(calculate_vif(X))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and fit the model
model = LinearRegression()
model.fit(X_train, y_train)

# Get coefficients and intercept
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)

# Predictions
y_pred = model.predict(X_test)

# Evaluation metrics

```

```

rmse = np.sqrt(mean_squared_error(y_test, y_pred)) r2 = r2_score(y_test, y_pred)
print(f"RMSE: {rmse}")
print(f"R^2: {r2}")
from sklearn.feature_selection import RFE

# Recursive Feature Elimination
rfe = RFE(estimator=LinearRegression(), n_features_to_select=5)

# Adjust features
rfe.fit(X_train, y_train)

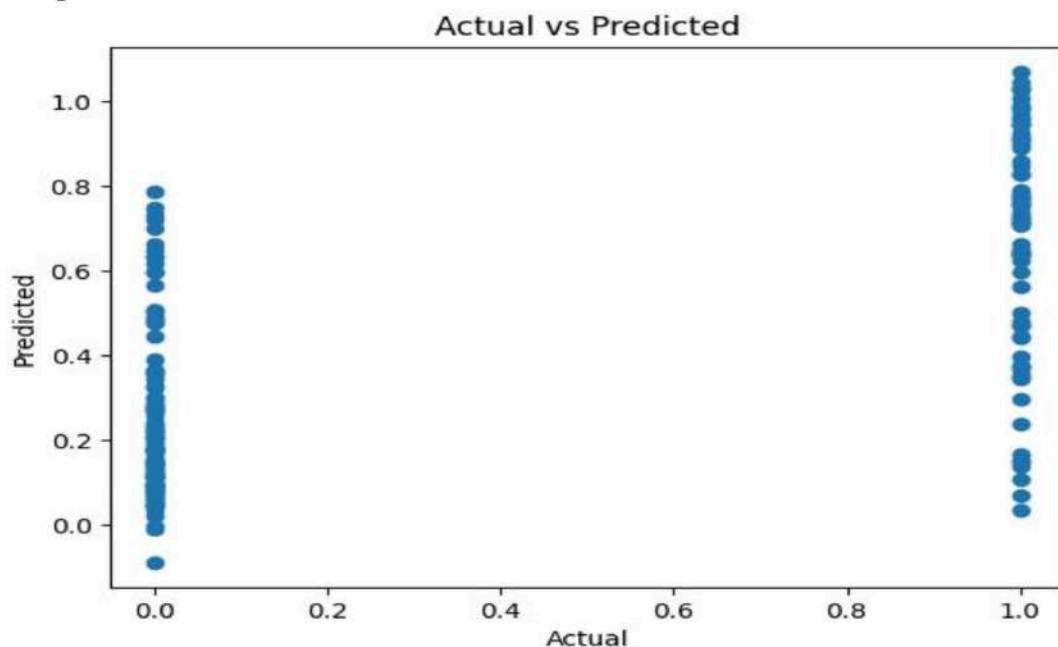
# Selected features
print("Selected Features:", X.columns[rfe.support_])

# Scatter plot of actual vs predicted values
plt.scatter(y_test, y_pred) plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Actual vs Predicted")
plt.show()

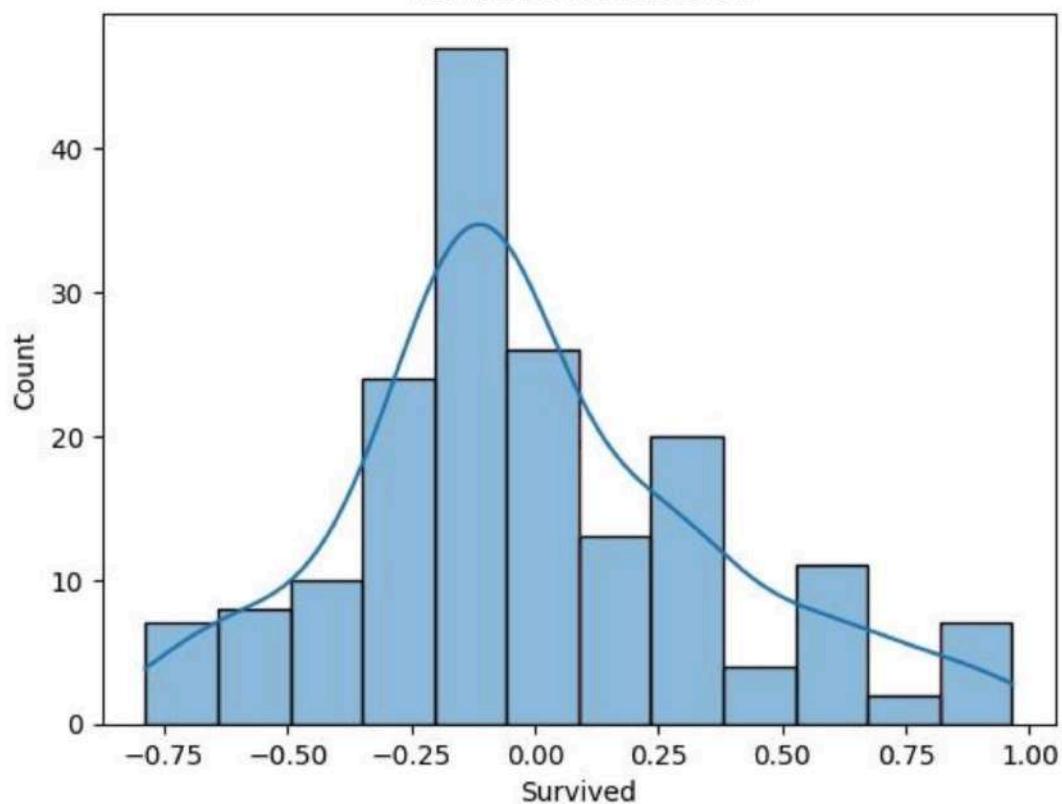
# Residuals
residuals = y_test - y_pred sns.histplot(residuals, kde=True)
plt.title("Residuals Distribution")
plt.show()

```

## **Output :**



Residuals Distribution



### **3c. Regularized Linear Models :**

Implement Regression variants like LASSO and Ridge on any generated dataset

#### **Code :**

##### **1. Set Up the Environment**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, Lasso, ElasticNet
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.datasets import make_regression
# Set random seed for reproducibility
np.random.seed(42)
```

##### **2. Generate a Synthetic Dataset**

###### **# Generate synthetic data**

```
X, y = make_regression(n_samples=1000,
```

###### **# Number of samples**

```
n_features=10,
```

###### **# Number of features**

```
noise=15,
```

###### **# Add some noise**

```
random_state=42
```

```
)
```

###### **# Convert to DataFrame for exploration**

```
data = pd.DataFrame(X, columns=[f"X{i}"
```

```
for i in range(1, 11)]) data["y"] = y
```

###### **# Display the first few rows**

```
print(data.head())
```

### 3. Split the Dataset

```
# Split data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(data.drop("y", axis=1),
```

```
# Features
```

```
data["y"],
```

```
# Target variable
```

```
test_size=0.2,
```

```
# 20% for testing
```

```
random_state=42
```

```
)
```

### 4. Train and Evaluate Ridge Regression

```
# Initialize Ridge Regression with a regularization parameter (alpha)
```

```
ridge = Ridge(alpha=1.0)
```

```
# Train the model
```

```
ridge.fit(X_train, y_train)
```

```
# Predictions
```

```
ridge_pred = ridge.predict(X_test)
```

```
# Evaluate Ridge Regression
```

```
ridge_rmse = np.sqrt(mean_squared_error(y_test, ridge_pred))
```

```
ridge_r2 = r2_score(y_test, ridge_pred)
```

```
print(f'Ridge RMSE: {ridge_rmse}')
```

```
print(f'Ridge R^2: {ridge_r2}')
```

### 5. Train and Evaluate Lasso Regression

```
# Initialize Lasso Regression
```

```
lasso = Lasso(alpha=0.1)
```

```
# Train the model
```

```
lasso.fit(X_train, y_train)
```

```
# Predictions  
lasso_pred = lasso.predict(X_test)
```

```
# Evaluate Lasso Regression  
lasso_rmse = np.sqrt(mean_squared_error(y_test, lasso_pred))  
lasso_r2 = r2_score(y_test, lasso_pred)  
print(f'Lasso RMSE: {lasso_rmse}')  
print(f'Lasso R^2: {lasso_r2}')  
# Features shrunk to  
zero print("Lasso Coefficients:", lasso.coef_)
```

## 6. Train and Evaluate ElasticNet Regression

```
# Initialize ElasticNet  
elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5) # l1_ratio balances L1 and L2 penalties
```

```
# Train the model  
elastic_net.fit(X_train, y_train)
```

```
# Predictions  
elastic_net_pred = elastic_net.predict(X_test)  
# Evaluate  
ElasticNet Regression elastic_net_rmse = np.sqrt(mean_squared_error(y_test,  
elastic_net_pred)) elastic_net_r2 = r2_score(y_test, elastic_net_pred)  
print(f'ElasticNet RMSE: {elastic_net_rmse}')  
print(f'ElasticNet R^2: {elastic_net_r2}')
```

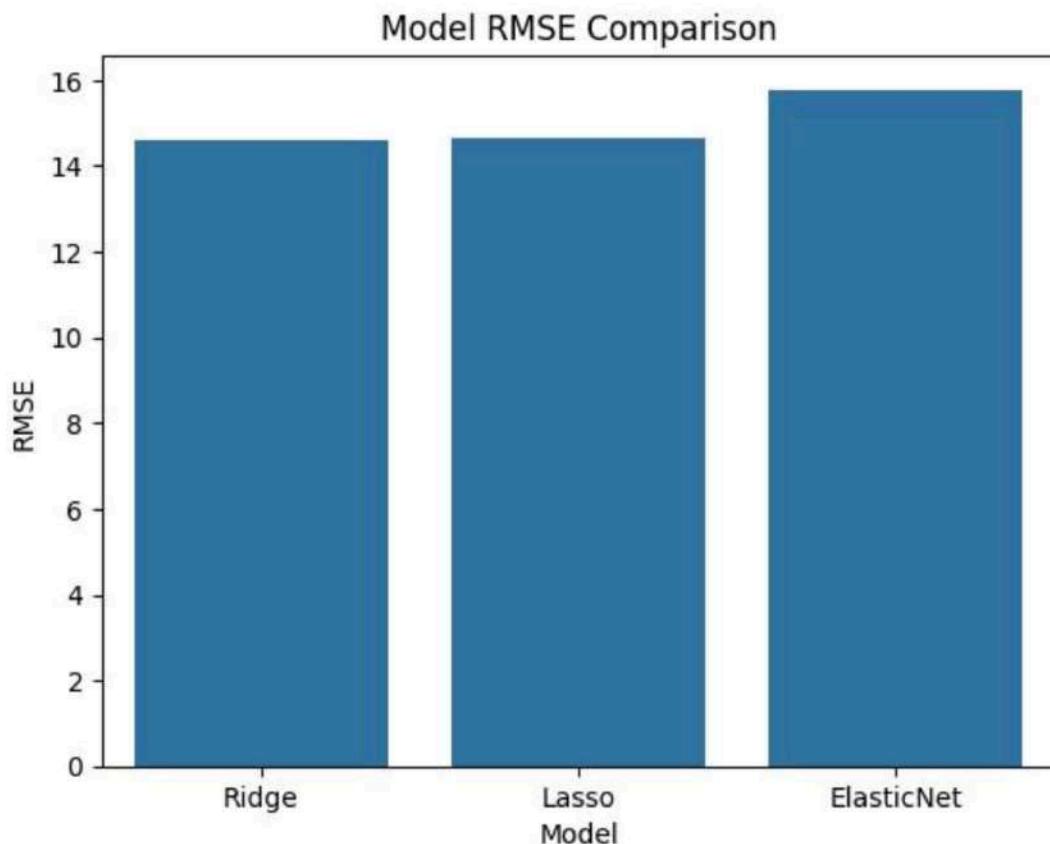
## 7. Compare Results

```
# Collect metrics  
metrics = pd.DataFrame({  
    "Model": ["Ridge", "Lasso", "ElasticNet"],  
    "RMSE": [ridge_rmse, lasso_rmse, elastic_net_rmse],  
    "R^2": [ridge_r2, lasso_r2, elastic_net_r2]})
```

```
})
print(metrics)

# Plot RMSE comparison
sns.barplot(data=metrics, x="Model", y="RMSE")
plt.title("Model RMSE Comparison")
plt.show()
```

### Output :



## **Practical 4 : Discriminative Models**

### **4a. Logistic Regression :**

Perform binary classification using logistic regression. Calculate accuracy, precision, recall, and understand the ROC curve."

#### **Code :**

##### **Step 1: Import Required Libraries**

###### **# Import necessary libraries**

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_curve, auc
import matplotlib.pyplot as plt
```

##### **Step 2: Prepare the Dataset**

```
from sklearn.datasets import make_classification
```

###### **# Create a synthetic dataset**

```
X, y = make_classification(n_samples=1000, n_features=10, n_classes=2, random_state=42)
```

###### **# Split data into training and testing sets**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

##### **Step 3: Train the Logistic Regression Model**

###### **# Initialize the logistic regression model**

```
logreg = LogisticRegression()
```

###### **# Train the model on the training data**

```
logreg.fit(X_train, y_train)
```

#### **Step 4: Make Predictions**

```
# Predict labels for the test set
```

```
y_pred = logreg.predict(X_test)
```

```
# Predict probabilities for the ROC curve
```

```
y_prob = logreg.predict_proba(X_test)[:, 1]
```

#### **Step 5: Evaluate the Model**

```
# Calculate metrics
```

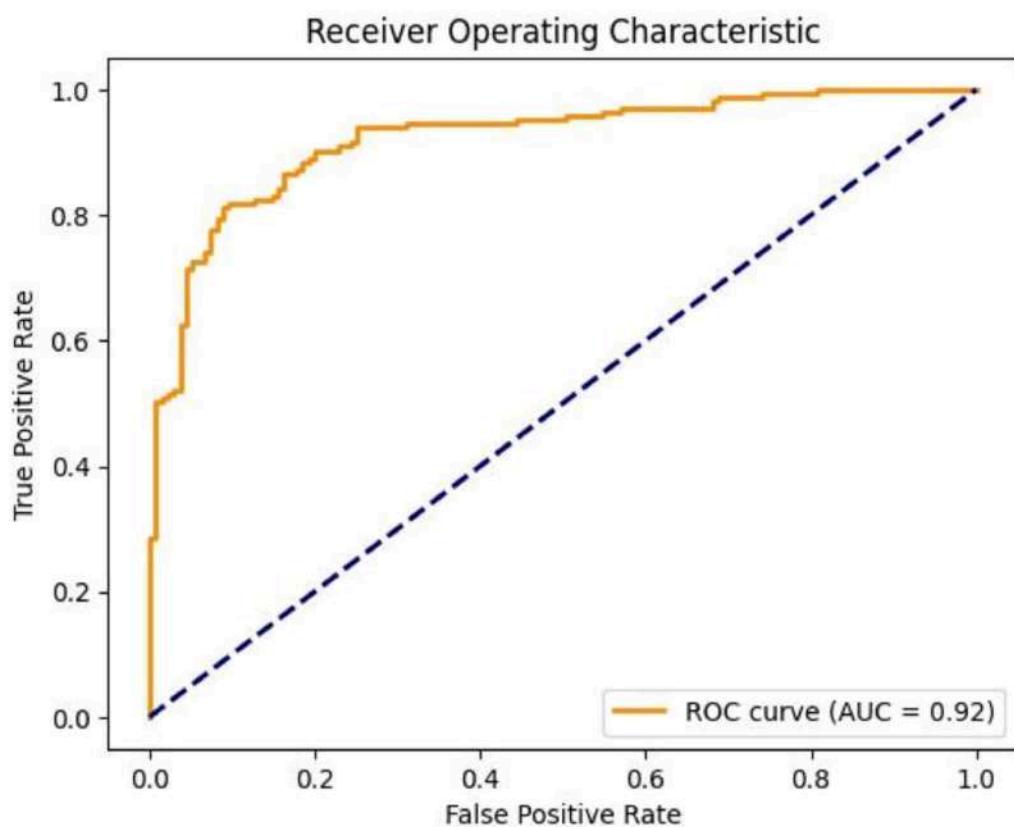
```
accuracy = accuracy_score(y_test, y_pred)
```

```
precision = precision_score(y_test, y_pred)
```

```
recall = recall_score(y_test, y_pred)
```

```
print(f'Accuracy: {accuracy:.2f}')
```

#### **Output :**



**4b .Implement and demonstrate k-nearest Neighbor algorithm. Read the training data from a .CSV file and build the model to classify a test sample. Print both correct and wrong predictions.**

**Code :**

```
Step 1: Import Required Libraries # Import necessary libraries
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

from google.colab import files
```

**Step 2: Create or Upload the CSV File**

```
# Check if the user wants to create a dataset or upload one
print("Do you have a CSV file to upload? (yes/no)")
response = input().lower()

if response == "yes":
    uploaded = files.upload()
    filename = list(uploaded.keys())[0]
else:

    # Create a synthetic dataset
    from sklearn.datasets import make_classification

    # Generate synthetic data
    X,y=make_classification(n_samples=200,n_features=5, n_classes=2, random_state=42)

    # Combine features and target into a single DataFrame
    data = pd.DataFrame(X, columns=[f"Feature_{i}" for i in range(X.shape[1])])
    data["Target"] = y

    # Save the dataset to a CSV file
    filename = "synthetic_data.csv"
    data.to_csv(filename, index=False)

    print(f"Synthetic dataset saved as {filename}.")
```

**Step 3: Load the CSV File into a DataFrame**

```
# Load the dataset into a DataFrame  
data = pd.read_csv(filename)  
# Display the first few rows of the dataset  
print("Loaded Dataset:")  
print(data.head())
```

**Step 4: Preprocess the Data****# Separate features (X) and labels (y)**

```
X = data.iloc[:, :-1].values # All columns except the last one  
y = data.iloc[:, -1].values # Last column as the target
```

**# Split the dataset into training and testing sets (80% train, 20% test)**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

**Step 5: Train the k-NN Model #****Initialize the k-NN model with k=3 knn**

```
= KNeighborsClassifier(n_neighbors=3) #
```

**Train the model on the training data**

```
knn.fit(X_train, y_train)
```

**Step 6: Predict Test Samples #****Predict the labels for the test set**

```
y_pred = knn.predict(X_test)
```

**Step 7: Evaluate and Print Predictions #****Calculate and display the accuracy**

```
accuracy = accuracy_score(y_test, y_pred)  
print(f"\nModel Accuracy:  
{accuracy:.2f}\n")
```

**# Display correct and incorrect predictions**

```
print("Correct Predictions:")  
for i in range(len(y_test)):  
    if y_pred[i] == y_test[i]:  
        print(f"Sample {i}: Predicted={y_pred[i]}, Actual={y_test[i]}")  
print("\nIncorrect Predictions:")  
  
for i in range(len(y_test)):
```

```
if y_pred[i] != y_test[i]:  
    print(f"Sample {i}: Predicted={y_pred[i]}, Actual={y_test[i]}")
```

## Output :

```
[ ] Model Accuracy: 0.88  
→ Correct Predictions:  
Sample 0: Predicted=0, Actual=0  
Sample 1: Predicted=1, Actual=1  
Sample 2: Predicted=1, Actual=1  
Sample 3: Predicted=0, Actual=0  
Sample 4: Predicted=1, Actual=1  
Sample 5: Predicted=1, Actual=1  
Sample 6: Predicted=0, Actual=0  
Sample 7: Predicted=0, Actual=0  
Sample 9: Predicted=1, Actual=1  
Sample 10: Predicted=1, Actual=1  
Sample 11: Predicted=1, Actual=1  
Sample 12: Predicted=0, Actual=0  
Sample 13: Predicted=0, Actual=0  
Sample 14: Predicted=0, Actual=0  
Sample 15: Predicted=0, Actual=0  
Sample 16: Predicted=0, Actual=0  
Sample 17: Predicted=1, Actual=1  
Sample 18: Predicted=1, Actual=1  
Sample 19: Predicted=0, Actual=0  
Sample 20: Predicted=0, Actual=0  
Sample 22: Predicted=1, Actual=1  
Sample 23: Predicted=1, Actual=1  
Sample 24: Predicted=1, Actual=1  
Sample 25: Predicted=1, Actual=1  
Sample 26: Predicted=1, Actual=1  
Sample 27: Predicted=0, Actual=0  
Sample 28: Predicted=0, Actual=0  
Sample 30: Predicted=1, Actual=1  
Sample 31: Predicted=1, Actual=1  
Sample 32: Predicted=1, Actual=1  
Sample 34: Predicted=0, Actual=0  
Sample 35: Predicted=1, Actual=1  
Sample 36: Predicted=1, Actual=1  
Sample 38: Predicted=1, Actual=1  
Sample 39: Predicted=1, Actual=1
```

```
Incorrect Predictions:  
Sample 8: Predicted=1, Actual=0  
Sample 21: Predicted=1, Actual=0  
Sample 29: Predicted=0, Actual=1  
Sample 33: Predicted=1, Actual=0  
Sample 37: Predicted=1, Actual=0
```

**4c. Build a decision tree classifier or regressor. Control hyperparameters like tree depth to avoid overfitting. Visualize the tree.**

**Code :**

**Step 1: Import Required Libraries**

**# Import necessary libraries**

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor,
plot_tree
from sklearn.metrics import accuracy_score, mean_squared_error
import matplotlib.pyplot as plt
from google.colab import files
```

**Step 2: Create or Upload the CSV File**

**# Check if the user wants to upload a file or generate one**

```
print("Do you have a CSV file to upload? (yes/no)")
```

```
response = input().lower()
```

```
if response == "yes":
```

**# Upload the CSV file**

```
uploaded = files.upload()
```

```
filename = list(uploaded.keys())[0]
```

```
else:
```

**# Generate synthetic data (classification or regression)**

```
from sklearn.datasets import make_classification, make_regression
print("Choose a task: (1) Classification (2) Regression")
```

```
task = int(input())
```

```
if task == 1:
```

**# Generate synthetic classification data**

```
X, y = make_classification(n_samples=200, n_features=5, random_state=42)
```

```
task_type = "classification"
```

```

else:
    # Generate synthetic regression data
    X, y = make_regression(n_samples=200, n_features=5, random_state=42)
    task_type = "regression"

    # Combine features and target into a single DataFrame
    data = pd.DataFrame(X, columns=[f"Feature_{i}" for i in range(X.shape[1])])
    data['Target'] = y

    # Save the dataset to a CSV file
    filename = "synthetic_data.csv"
    data.to_csv(filename, index=False)
    print(f"Synthetic {task_type} dataset saved as {filename}.")

```

### **Step 3: Load the Dataset**

```

# Load the dataset
data = pd.read_csv(filename)

# Display the first few rows of the dataset
print("Dataset Preview:")
print(data.head())

```

### **Step 4: Preprocess the Data**

```

# Separate features and target
X = data.iloc[:, :-1].values # All columns except the last one
y = data.iloc[:, -1].values # Last column as the target

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

### **Step 5: Build the Decision Tree**

```

# Define the tree depth to avoid overfitting
max_depth = 3

```

```

# Initialize the model
if task_type == "classification":
    model = DecisionTreeClassifier(max_depth=max_depth, random_state=42)
else:
    model = DecisionTreeRegressor(max_depth=max_depth, random_state=42)

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

```

### **Step 6: Make Predictions**

```

# Predict on the test set
y_pred = model.predict(X_test)
# Evaluate the model
if task_type == "classification":
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.2f}")
else:
    mse = mean_squared_error(y_test, y_pred)
    print(f"Mean Squared Error: {mse:.2f}")

```

### **Step 7: Visualize the Tree**

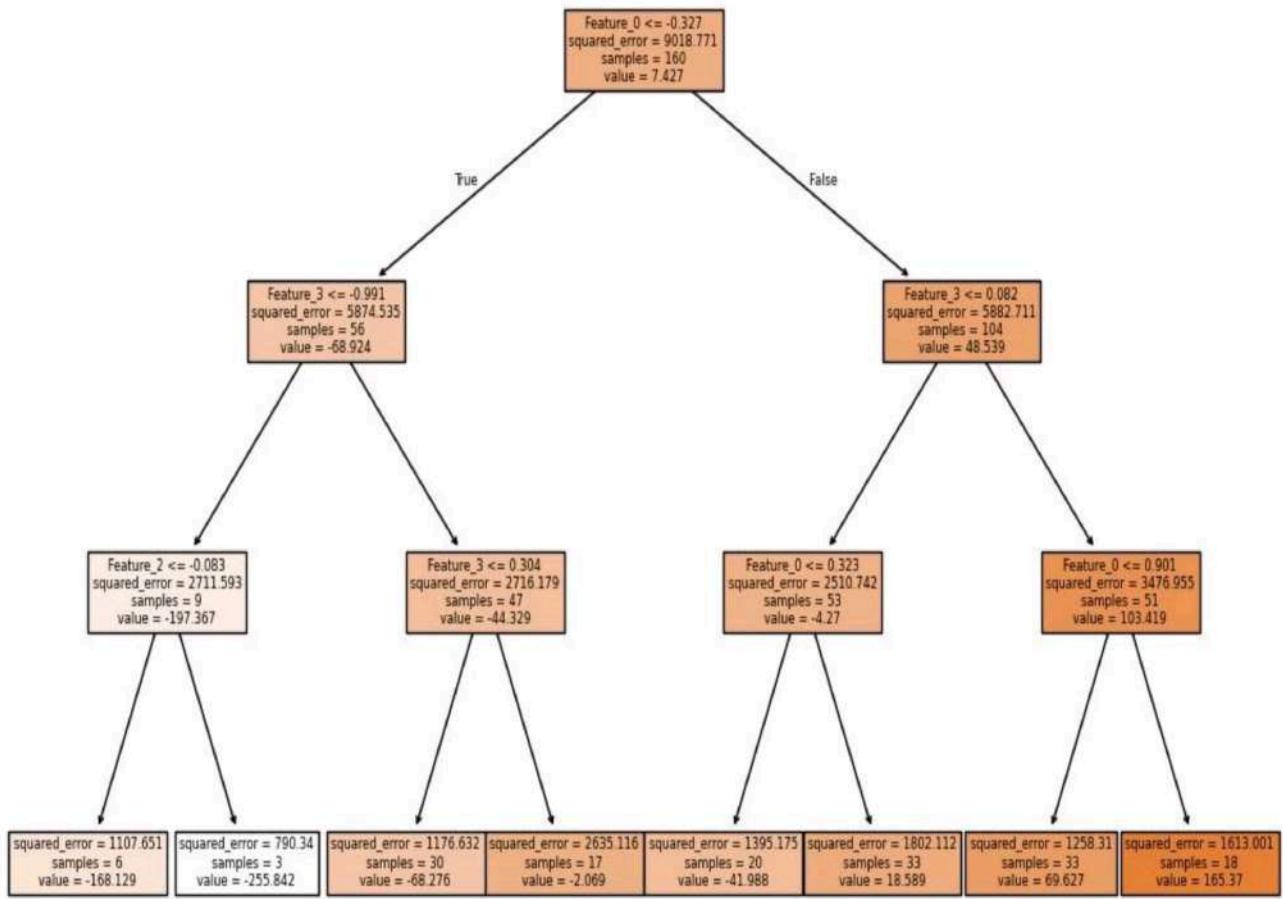
```

# Visualize the decision tree
plt.figure(figsize=(12, 8))
plot_tree(model, feature_names=data.columns[:-1], class_names=str(np.unique(y)))
if task_type == "classification" else None, filled=True)
plt.title("Decision Tree Visualization")
plt.show()

```

### **Output :**

## Decision Tree Visualization



#### **4d. Implement a Support Vector Machine for any relevant dataset.**

##### **Code:**

###### **Step 1: Import Required Libraries**

```
# Import necessary libraries
```

```
import pandas as pd
```

```
import numpy as np
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.svm import SVC
```

```
from sklearn.metrics import accuracy_score, classification_report
```

```
from google.colab import files
```

###### **Step 2: Create or Upload a Dataset**

###### **# Check if the user wants to upload a file or generate one**

```
print("Do you have a CSV file to upload? (yes/no)")
```

```
response = input().lower()
```

```
if response == "yes":
```

###### **# Upload the CSV file**

```
uploaded = files.upload()
```

```
filename = list(uploaded.keys())[0]
```

```
else:
```

###### **# Generate synthetic classification data**

```
from sklearn.datasets import make_classification
```

```
X, y = make_classification(n_samples=200, n_features=5, n_classes=2, random_state=42)
```

###### **# Combine features and target into a DataFrame**

```
data = pd.DataFrame(X, columns=[f"Feature_{i}"
```

```
for i in range(X.shape[1])])
```

```
data['Target'] = y
```

###### **#Save the synthetic dataset to a CSV file**

```
filename="synthetic_data.csv"
data.to_csv(filename,index=False)
print(f"Synthetic dataset saved as {filename}.")
```

### **Step 3: Load the Dataset**

```
# Load the dataset into a DataFrame
```

```
data = pd.read_csv(filename)
```

```
# Display the first few rows of the dataset
```

```
print("Dataset Preview:")
```

```
print(data.head())
```

### **Step 4: Preprocess the Data**

```
# Separate features (X) and target (y)
```

```
X = data.iloc[:, :-1].values # All columns except the last one
```

```
y = data.iloc[:, -1].values # Last column as the target
```

```
# Split the dataset into training (80%) and testing (20%) sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

### **Step 5: Train the Support Vector Machine**

```
# Initialize the SVM model (use RBF kernel as default)
```

```
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
```

```
# Train the SVM model on the training data
```

```
svm_model.fit(X_train, y_train)
```

### **Step 6: Make Predictions**

```
# Predict the labels for the test set
```

```
y_pred = svm_model.predict(X_test)
```

### **Step 7: Evaluate the Model**

```
# Calculate and print the accuracy
```

```
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f"Model Accuracy: {accuracy:.2f}")
```

```
# Print a detailed classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

## Step 8: Visualize the Decision Boundary (Optional for 2D Data)

```
import matplotlib.pyplot as plt

# Generate 2D synthetic data
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=100, centers=2, random_state=42, cluster_std=1.5)

# Fit the SVM on this data
svm_model.fit(X, y)

# Plot the decision boundary
plt.figure(figsize=(8, 6))

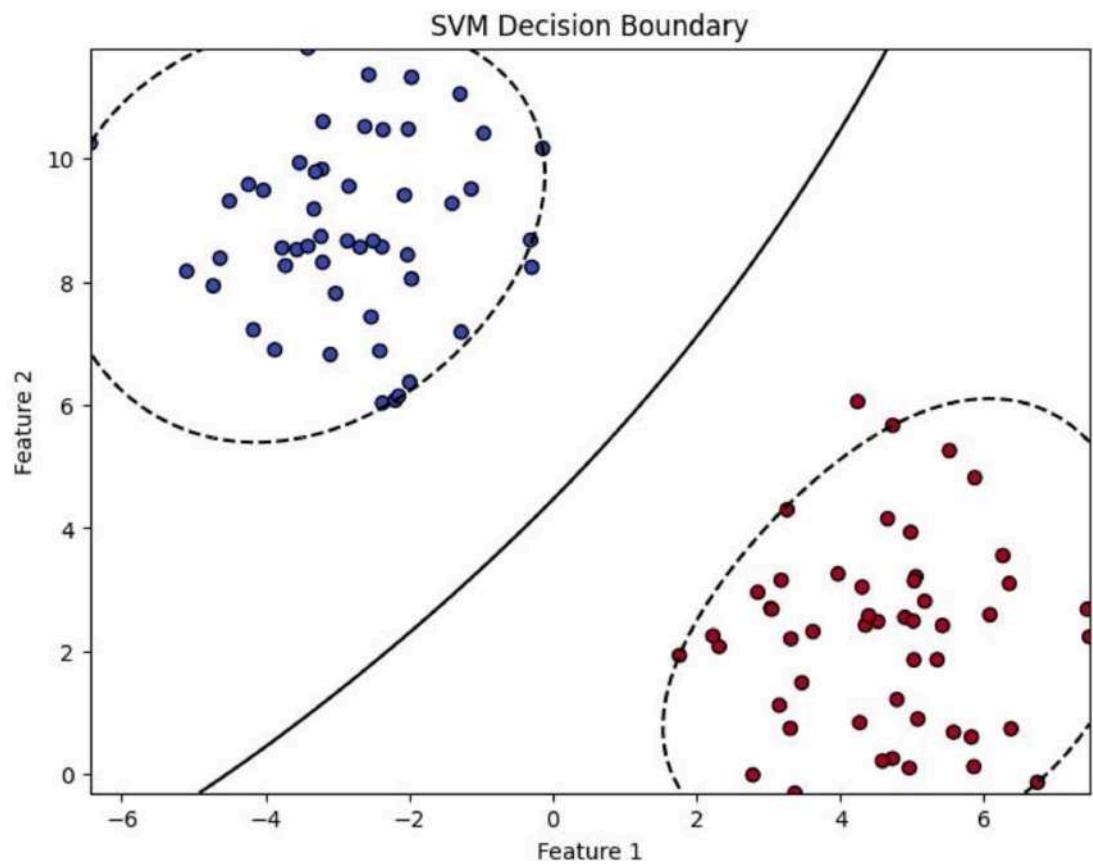
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='coolwarm', edgecolor='k')
# Create a grid to evaluate the model

xx, yy = np.meshgrid(np.linspace(X[:, 0].min(), X[:, 0].max(), 100), np.linspace(X[:, 1].min(), X[:, 1].max(), 100))
Z = svm_model.decision_function(np.c_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

# Plot the decision boundary and margins
plt.contour(xx, yy, Z, levels=[-1, 0, 1], linestyles=['--', ':', '-'], colors='k')
plt.title("SVM Decision Boundary")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

## Output :



#### **4e. Train a random forest ensemble. Experiment with the number of trees and feature sampling. Compare performance to a single decision tree.**

**Code :**

**Step 1: Import Required Libraries # Import necessary libraries**

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report

from google.colab import files
```

**Step 2: Create or Upload a Dataset**

```
# Check if the user wants to upload a file or generate one
print("Do you have a CSV file to upload? (yes/no)")
response = input().lower()
if response == "yes":
    # Upload the CSV file
    uploaded = files.upload()
    filename = list(uploaded.keys())[0]
else:
    # Generate synthetic classification data
    from sklearn.datasets import make_classification
    X, y = make_classification(n_samples=300, n_features=10, n_classes=2, random_state=42)
    # Combine features and target into a DataFrame
    data = pd.DataFrame(X, columns=[f'Feature_{i}' for i in range(X.shape[1])])
    data['Target'] = y
# Save the synthetic dataset to a CSV file
filename = "synthetic_data.csv"
data.to_csv(filename, index=False)
print(f'Synthetic dataset saved as {filename}.')
```

**Step 3: Load the Dataset**

```
# Load the dataset
```

```
data = pd.read_csv(filename)

# Display the first few rows of the dataset
print("Dataset Preview:")
print(data.head())
```

#### **Step 4: Preprocess the Data**

```
# Separate features (X) and target (y)
```

```
X = data.iloc[:, :-1].values # All columns except the last one
y = data.iloc[:, -1].values # Last column as the target
```

```
# Split the dataset into training (80%) and testing (20%) sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

#### **Step 5: Train a Single Decision Tree Classifier**

```
# Initialize and train the Decision Tree model
```

```
decision_tree = DecisionTreeClassifier(random_state=42)
decision_tree.fit(X_train, y_train)
```

```
# Predict and evaluate
```

```
y_pred_tree = decision_tree.predict(X_test)
accuracy_tree = accuracy_score(y_test, y_pred_tree)
print(f"Decision Tree Accuracy: {accuracy_tree:.2f}")
```

#### **Step 6: Train a Random Forest Classifier**

```
# Initialize the Random Forest model with hyperparameter tuning
```

```
random_forest = RandomForestClassifier(n_estimators=100, max_features='sqrt',
random_state=42)
```

```
# Train the model
```

```
random_forest.fit(X_train, y_train)
```

```
# Predict and evaluate
```

```
y_pred_rf = random_forest.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f"Random Forest Accuracy (100 trees, sqrt features): {accuracy_rf:.2f}")
```

#### **Step 7: Experiment with Random Forest Hyperparameters**

```
# Experiment with fewer trees and different feature sampling
```

```
rf_experiment = RandomForestClassifier(n_estimators=50, max_features=3,
random_state=42)
```

```
rf_experiment.fit(X_train, y_train)
```

## # Predict and evaluate

```
y_pred_rf_exp = rf_experiment.predict(X_test)  
accuracy_rf_exp = accuracy_score(y_test, y_pred_rf_exp)  
print(f"Random Forest Accuracy (50 trees, max_features=3): {accuracy_rf_exp:.2f}")
```

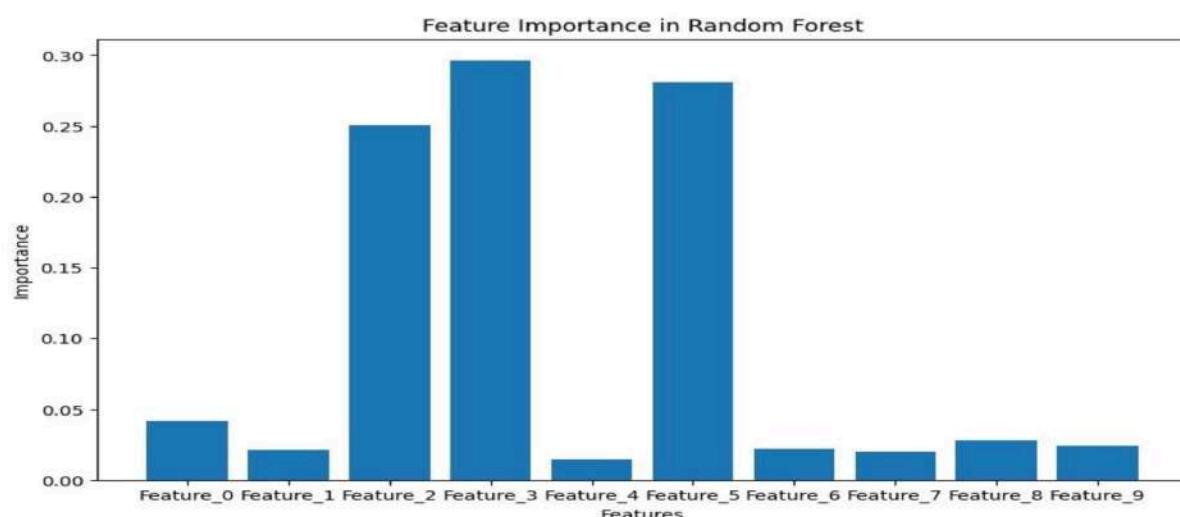
## Step 8: Compare the Models

```
print("\nModel Comparison:")  
print(f"Decision Tree Accuracy: {accuracy_tree:.2f}")  
print(f"Random Forest Accuracy (100 trees): {accuracy_rf:.2f}")  
print(f"Random Forest Accuracy (50 trees, max_features=3): {accuracy_rf_exp:.2f}")
```

## Step 9: Visualize Feature Importance (Optional)

```
import matplotlib.pyplot as plt  
  
# Extract feature importance from the Random Forest model  
feature_importances = random_forest.feature_importances_  
  
# Plot the feature importance  
plt.figure(figsize=(10, 6))  
plt.bar(range(len(feature_importances)), feature_importances, tick_label=data.columns[:-1])  
plt.title("Feature Importance in Random Forest")  
plt.xlabel("Features")  
plt.ylabel("Importance")  
plt.show()
```

## Output :



#### **4f. Implement a gradient boosting machine (e.g., XGBoost). Tune hyperparameters and explore feature importance.**

**Code :**

##### **Step 1: Import Required Libraries**

**# Import necessary libraries**

```
import pandas as pd
```

```
import numpy as np
```

```
from sklearn.model_selection import train_test_split, GridSearchCV  
from sklearn.metrics import accuracy_score, classification_report  
from xgboost import XGBClassifier, plot_importance
```

```
import matplotlib.pyplot as plt
```

```
from google.colab import files
```

##### **Step 2: Create or Upload a Dataset**

**# Check if the user wants to upload a file or generate**

```
one print("Do you have a CSV file to upload? (yes/no)")
```

```
response = input().lower()
```

```
if response == "yes":
```

```
# Upload the CSV file
```

```
uploaded=files.upload()
```

```
filename = list(uploaded.keys())[0]
```

```
else:
```

**# Generate synthetic classification data**

```
from sklearn.datasets import make_classification
```

```
X, y = make_classification(n_samples=300, n_features=10, n_classes=2, random_state=42)
```

**# Combine features and target into a DataFrame**

```
data = pd.DataFrame(X, columns=[f"Feature_{i}"
```

```
for i in range(X.shape[1])])data['Target'] = y
```

**# Save the synthetic dataset to a CSV file**

```
filename="synthetic_data.csv"
data.to_csv(filename,index=False)
print(f"Synthetic dataset saved as {filename}.")
```

### **Step 3: Load the Dataset**

#### **# Load the dataset**

```
data = pd.read_csv(filename)
# Display the first few rows of the
dataset print("Dataset Preview:")
print(data.head())
```

### **Step 4: Preprocess the Data**

#### **# Separate features (X) and target (y)**

```
X = data.iloc[:, :-1].values # All columns except the last one
y = data.iloc[:, -1].values # Last column as the target
```

#### **# Split the dataset into training (80%) and testing (20%) sets**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

### **Step 5: Train a Basic XGBoost Model**

#### **# Initialize and train the XGBoost model with default parameters**

```
xgb = XGBClassifier(random_state=42)
xgb.fit(X_train, y_train)
```

#### **# Predict and evaluate the model**

```
y_pred = xgb.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"XGBoost Accuracy (Default Parameters): {accuracy:.2f}")
```

### **Step 6: Tune Hyperparameters with GridSearchCV**

#### **# Define a grid of hyperparameters**

```
param_grid = { 'n_estimators': [50, 100, 150], 'learning_rate': [0.01, 0.1, 0.2], 'max_depth': [3, 5, 7] }
```

#### **# Initialize GridSearchCV**

```
grid_search =
GridSearchCV(estimator=XGBClassifier(random_state=42), param_grid=param_grid,
scoring='accuracy', cv=3, verbose=1)
# Fit the model with grid search
grid_search.fit(X_train, y_train)
```

```

# Best parameters from GridSearch
print(f'Best Parameters: {grid_search.best_params_}')
# Train the final model with the best parameters
best_xgb = grid_search.best_estimator_
# Predict and evaluate

y_pred_best = best_xgb.predict(X_test)

accuracy_best = accuracy_score(y_test, y_pred_best)
print(f'XGBoost Accuracy (Tuned Parameters): {accuracy_best:.2f}')

```

### **Step 7: Explore Feature Importance**

**# Plot feature importance for the tuned model**

```

plt.figure(figsize=(10, 6))

plot_importance(best_xgb, importance_type='weight', xlabel="Importance",
                ylabel="Features")

plt.title("XGBoost Feature Importance")
plt.show()

```

### **Step 8: Evaluate the Model**

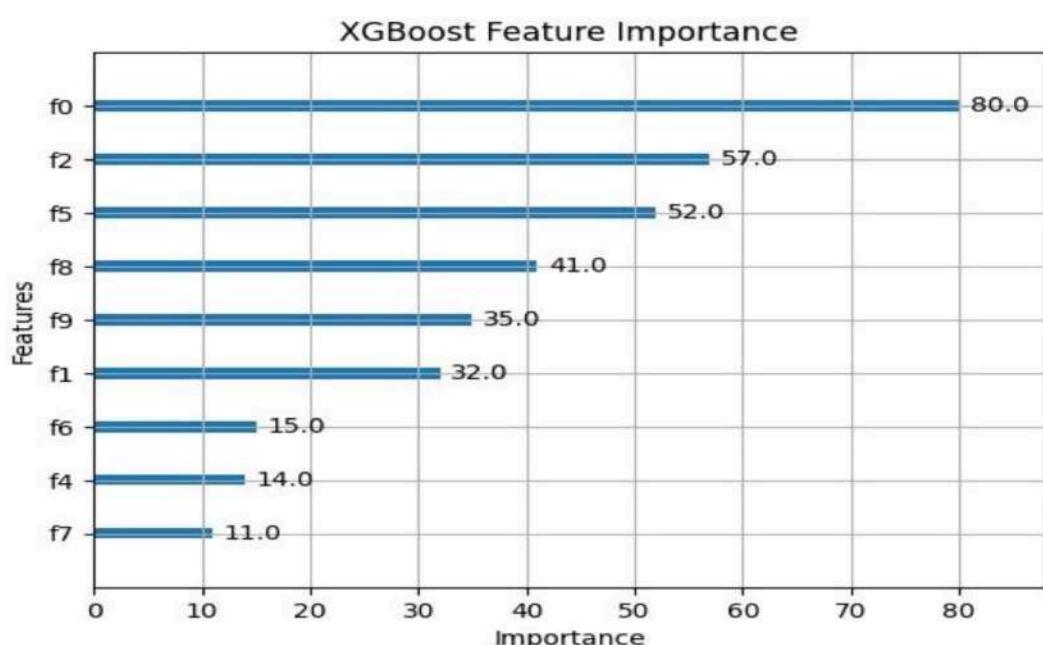
**# Print a detailed classification report**

```

print("Classification Report:")
print(classification_report(y_test, y_pred_best))

```

**Output :**



## Practical 5

**5a. Implement and demonstrate the working of a Naive Bayesian classifier using a sample data set. Build the model to classify a test sample.**

### **Step 1: Import Required Libraries**

#### **# Import necessary libraries**

```
import pandas as pd  
import numpy as np  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import accuracy_score,classification_report  
from sklearn.naive_bayes import GaussianNB  
from google.colab import files
```

### **Step 2: Create or Upload a Dataset**

#### **# Ask if the user wants to upload a file or generate one**

```
print("Do you have a CSV file to upload? (yes/no)")
```

```
response = input().lower()
```

```
if response == "yes":
```

#### **# Upload the CSV file**

```
uploaded = files.upload()
```

```
filename = list(uploaded.keys())[0]
```

```
else:
```

#### **# Generate synthetic classification data**

```
from sklearn.datasets import make_classification
```

```
X, y = make_classification(n_samples=300, n_features=8, n_classes=2, random_state=42)
```

#### **# Combine features and target into a DataFrame**

```
data = pd.DataFrame(X, columns=[f'Feature_{i}' for i in range(X.shape[1])])  
data['Target'] = y
```

#### **# Save the synthetic dataset to a CSV file**

```
filename = "synthetic_naive_bayes_data.csv"
```

```
data.to_csv(filename, index=False)
```

```
print(f'Synthetic dataset saved as {filename}.')
```

### **Step 3: Load the Dataset**

```
# Load the dataset  
data = pd.read_csv(filename)  
  
# Display the first few rows of the dataset  
print("Dataset Preview:")  
print(data.head())
```

### **Step 4: Preprocess the Data**

```
# Separate features (X) and target (y)  
X = data.iloc[:, :-1].values # All columns except the last one  
y = data.iloc[:, -1].values # Last column as the target  
  
# Split the dataset into training (80%) and testing (20%) sets  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

### **Step 5: Train a Naive Bayes Classifier**

```
# Initialize the Gaussian Naive Bayes classifier  
naive_bayes = GaussianNB()  
  
# Train the model  
naive_bayes.fit(X_train, y_train)
```

### **Step 6: Make Predictions and Evaluate**

```
# Predict on the test set  
y_pred = naive_bayes.predict(X_test)  
  
# Evaluate the model  
accuracy = accuracy_score(y_test, y_pred)  
print(f'Naive Bayes Accuracy:  
{accuracy:.2f}')  
  
# Detailed classification report  
print("Classification Report:")  
print(classification_report(y_test, y_pred))
```

### **Step 7: Test the Model with a Custom Sample**

```

# Define a sample test input (replace with meaningful values based on your dataset)
test_sample = [X_test[0]]

# Taking the first test sample for demonstration

# Predict the class for the test sample

predicted_class = naive_bayes.predict(test_sample)

print(f"Test Sample: {test_sample}")

print(f"Predicted Class: {predicted_class[0]}")

```

## Output :

Dataset Preview:						
	Feature_0	Feature_1	Feature_2	Feature_3	Feature_4	Feature_5
0	-1.274158	1.317988	-2.423879	0.906946	-1.583903	-0.331811
1	1.607963	-1.649959	0.299293	-0.891720	1.301741	1.508502
2	-0.154167	0.161033	2.210523	0.139400	-0.557492	0.087713
3	-0.920991	0.949136	-1.613561	0.588410	1.471170	-0.529287
4	1.013304	-1.038578	-0.305225	-0.539334	-0.609512	1.048078
	Feature_6	Feature_7	Target			
0	-0.452306	0.760415	1			
1	0.742095	1.561511	0			
2	0.963879	-1.369803	0			
3	-1.371901	-0.209324	0			
4	-1.065114	-0.186971	0			

```

→ Test Sample: [array([-0.90320608,  0.9220511 , -1.32308979,  0.41081065,  1.64201516,
   -1.23559176, -0.63896175,  1.00981709])]

Predicted Class: 1

```

## **5b. Implement Hidden Markov Models using hmmlearn**

**Code :**

**Step 1: Install Required Libraries**

```
# Install hmmlearn
```

```
!pip install hmmlearn
```

**Step 2: Import Required Libraries**

```
# Import necessary libraries
```

```
import numpy as np
```

```
import pandas as pd
```

```
from hmmlearn import hmm
```

```
import matplotlib.pyplot as plt
```

**Step 3: Create or Load a Dataset**

```
# Generate synthetic observable data
```

```
np.random.seed(42)
```

```
# Create a sequence of observations and hidden states
```

```
observations = np.random.choice(['A', 'B', 'C'], size=100, p=[0.5, 0.3, 0.2])
```

```
hidden_states = np.random.choice(['X', 'Y'], size=100, p=[0.6, 0.4])
```

```
# Save the data in a DataFrame for analysis
```

```
data = pd.DataFrame({'Observations': observations, 'Hidden States': hidden_states})
```

```
print("Generated Data:")
```

```
print(data.head())
```

**Step 4: Encode Observations**

```
# Encode the observations into integers
```

```
observation_mapping = {obs: idx for idx, obs in enumerate(np.unique(observations))}
```

```
encoded_observations = np.array([observation_mapping[obs] for obs in observations])
```

```
# Print the mapping
```

```
print("Observation Encoding:")
```

```
print(observation_mapping)
```

## **Step 5: Initialize and Configure the HMM**

**# Initialize the HMM model**

```
n_states = 2 # Number of hidden states
```

```
n_observations = len(observation_mapping)
```

**# Number of unique observations**

```
model = hmm.MultinomialHMM(n_components=n_states, random_state=42, n_iter=100, tol=0.01)
```

**# Define start probabilities (initial distribution of states)**

```
start_probs = np.array([0.6, 0.4]) # Assumed probabilities
```

```
model.startprob_ = start_probs
```

**# Define transition probabilities between states**

```
trans_probs = np.array([
```

```
    [0.7, 0.3], # From state X
```

```
    [0.4, 0.6], # From state Y])
```

```
model.transmat_ = trans_probs
```

**# Define emission probabilities (probability of observations given states)**

```
emission_probs = np.array([
```

```
    [0.5, 0.4, 0.1], # State X emits A, B, C
```

```
    [0.2, 0.3, 0.5], # State Y emits A, B, C
```

```
])
```

```
model.emissionprob_ = emission_probs
```

**# Print the configured model parameters**

```
print("Start Probabilities:", model.startprob_)
```

```
print("Transition Matrix:", model.transmat_)
```

```
print("Emission Probabilities:",
```

```
model.emissionprob_)
```

## **Step 6: Train the Model**

**# Reshape the data for HMM (requires 2D array)**

```
encoded_observations = encoded_observations.reshape(-1, 1)
```

**# Fit the model**

```
model.fit(encoded_observations)
```

**# Predict hidden states for the observations**

```
predicted_states = model.predict(encoded_observations)
```

```
# Print the predicted states
print("Predicted States:")
print(predicted_states)
```

### Step 7: Visualize the Results

```
# Map predicted states back to their original labels
```

```
state_mapping = {0: 'X', 1: 'Y'}
```

```
predicted_state_labels = [state_mapping[state] for state in predicted_states]
```

```
# Add predicted states to the DataFrame
```

```
data['Predicted States'] = predicted_state_labels
```

```
# Display the first few rows with predicted states
```

```
print("Data with Predicted States:")
```

```
print(data.head())
```

```
# Plot the observations and predicted states
```

```
plt.figure(figsize=(12, 6))
```

```
plt.plot(data['Observations'], label='Observations', marker='o', linestyle='-', alpha=0.7)
```

```
plt.plot(data['Predicted States'], label='Predicted States', marker='x', linestyle='--', alpha=0.7)
```

```
plt.legend()
```

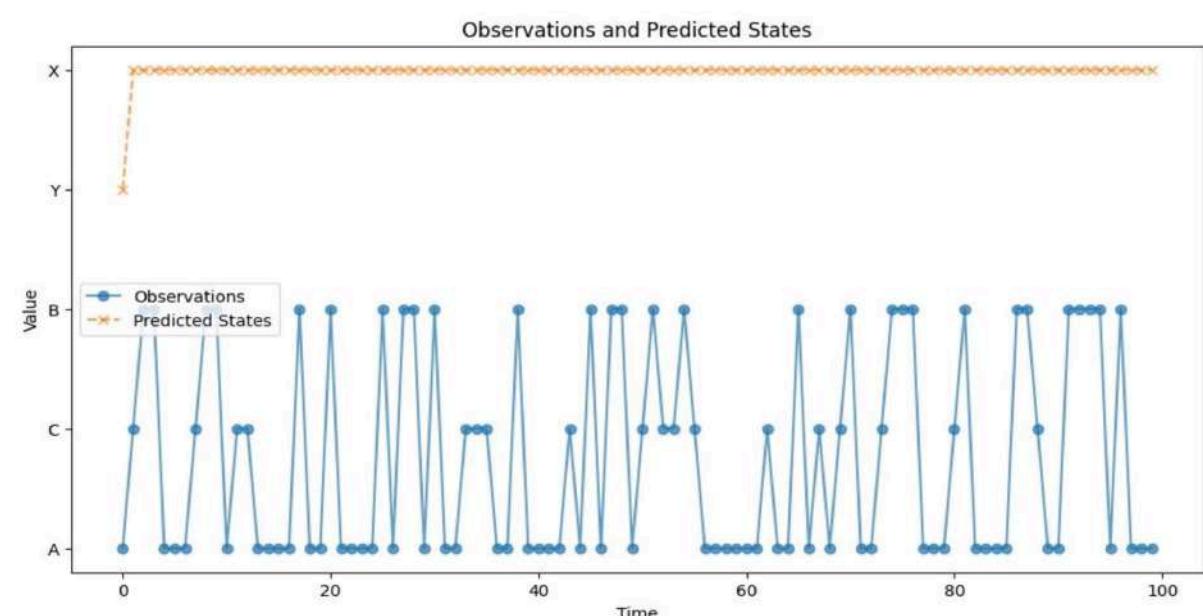
```
plt.title("Observations and Predicted States")
```

```
plt.xlabel("Time")
```

```
plt.ylabel("Value")
```

```
plt.show()
```

### Output :



## **Practical 6 : Probabilistic Model**

### **6a. Implement Bayesian Linear Regression to explore prior and posterior distribution.**

Bayesian Linear Regression is a probabilistic approach to linear regression that incorporates uncertainty in the model parameters. Instead of estimating point values for parameters (as in traditional linear regression), we estimate distributions over the parameters.

#### **Code :**

##### **Step 1: Install Required Libraries**

###### **# Install necessary libraries**

```
!pip install matplotlib seaborn scikit-learn
```

##### **Step 2: Import Required Libraries**

###### **# Import necessary libraries**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import BayesianRidge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from google.colab import files
```

##### **Step 3: Create or Upload a Dataset**

###### **# Upload a CSV file if you have one**

```
print("Do you have a CSV file to upload? (yes/no)")
response = input().lower()
if response == "yes":
    # Upload the CSV file
    uploaded = files.upload()
```

```

filename = list(uploaded.keys())[0]
else:

# Generate synthetic data for demonstration
np.random.seed(42)

X = np.random.rand(100, 1) * 10

# Random data between 0 and 10

y = 2 * X + 1 + np.random.randn(100, 1) * 2

# y = 2x + 1 with some noise

# Convert to a DataFrame

data = pd.DataFrame(np.hstack((X, y)), columns=["X", "y"])

# Save to CSV for convenience

filename="synthetic_data.csv"

data.to_csv(filename,index=False)

print(f"Synthetic dataset saved as {filename}.")

```

#### **Step 4: Load and Explore the Data**

```

# Load the dataset (for CSV file)
data = pd.read_csv(filename)

# Display first few rows
print("Dataset Preview:")
print(data.head())

```

#### **Step 5: Preprocess the Data**

```

# Separate features (X) and target (y)

X = data["X"].values.reshape(-1, 1) # Feature matrix
y = data["y"].values # Target vector

# Split the dataset into training (80%) and testing (20%) sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

#### **Step 6: Implement Bayesian Linear Regression Model**

```

# Initialize the BayesianRidge model (which implements Bayesian Linear Regression)

bayesian_regressor = BayesianRidge()

# Fit the model on the training data

bayesian_regressor.fit(X_train, y_train)

```

```
# Predict on the test data  
y_pred = bayesian_regressor.predict(X_test)
```

### Step 7: Visualize the Prior and Posterior Distributions

```
# Plot the prior and posterior distributions of the parameters  
  
fig, ax = plt.subplots(1, 2, figsize=(12, 6))  
  
# Plot prior distribution (assuming the model starts with a standard prior)  
ax[0].set_title("Prior Distribution (Assumed)") ax[0].hist(np.random.normal(0, 1, 1000),  
bins=50, alpha=0.7, color='blue', label="Prior") ax[0].legend()  
  
# Plot posterior distribution (after model fitting)  
  
ax[1].set_title("Posterior Distribution (After Fitting)")  
ax[1].hist(bayesian_regressor.coef_, bins=50, alpha=0.7, color='green',  
label="Posterior") ax[1].legend()  
  
plt.show()
```

### Step 8: Evaluate the Model Performance #

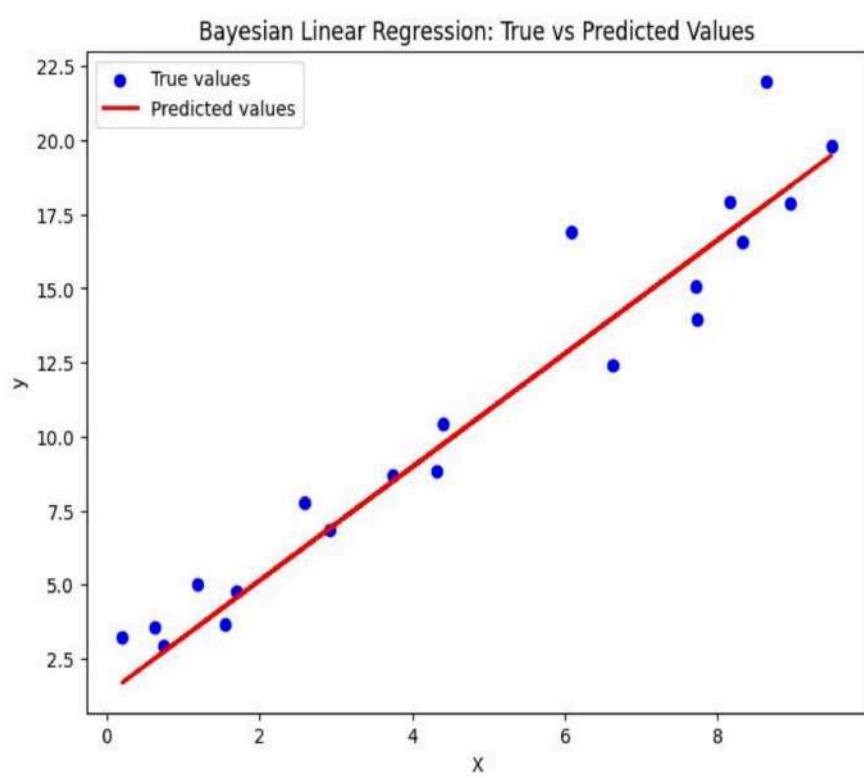
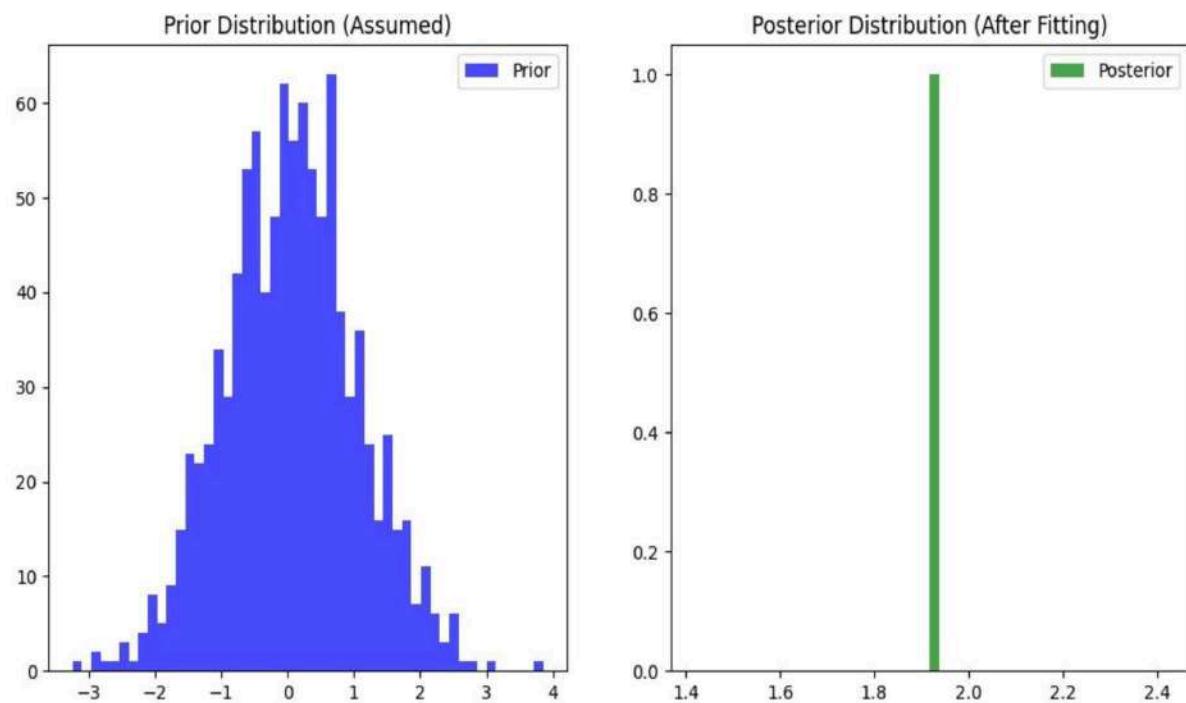
```
Calculate the Mean Squared Error (MSE)  
mse = mean_squared_error(y_test, y_pred)  
print(f'Mean Squared Error (MSE):  
{mse:.2f}')
```

### Step 9: Visualize the Fit of the Model

```
# Plot the true values and the predicted values  
  
plt.figure(figsize=(8, 6))  
  
plt.scatter(X_test, y_test, color="blue", label="True values")  
plt.plot(X_test, y_pred, color="red", label="Predicted values",  
linewidth=2)  
  
plt.title("Bayesian Linear Regression: True vs Predicted Values")  
plt.xlabel("X")  
plt.ylabel("y")  
plt.legend()  
plt.show()
```

**Mean Squared Error (MSE): 3.9**

## Output :



## **6b. Implement Gaussian Mixture Models for density estimation and unsupervised clustering.**

**Code :**

### **Step 1: Install Required Libraries**

**# Install required libraries**

```
!pip install matplotlib seaborn scikit-learn
```

### **Step 2: Import Required Libraries**

**# Import necessary libraries**

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
from sklearn.mixture import GaussianMixture
```

```
from sklearn.model_selection import train_test_split
```

```
from google.colab import files
```

### **Step 3: Create or Upload a Dataset**

**#Ask if the user has a CSV file to upload**

```
print("Do you have a CSV file to upload? (yes/no)")
```

```
response = input().lower()
```

```
if response == "yes":
```

**# Upload the CSV file**

```
uploaded = files.upload()
```

```
filename = list(uploaded.keys())[0]
```

```
else:
```

**# Generate synthetic 2D data with two clusters for demonstration**

```
np.random.seed(42)
```

**# Generate data for two Gaussian distributions**

```
X1 = np.random.normal(loc=0, scale=1, size=(300, 2)) # Cluster 1: mean = 0, std = 1
```

```
X2 = np.random.normal(loc=5, scale=1, size=(300, 2)) # Cluster 2: mean = 5, std = 1 #
```

**Stack the data to create a dataset**

```
X = np.vstack([X1, X2])
```

```
# Create DataFrame to simulate the CSV file for consistency
data = pd.DataFrame(X, columns=["Feature_1", "Feature_2"])
filename = "synthetic_data.csv"
data.to_csv(filename, index=False)
print(f"Synthetic dataset saved as {filename}.")
```

#### Step 4: Load and Explore the Dataset

```
# Load the dataset (if CSV file is uploaded)
data = pd.read_csv(filename)
# Display the first few rows
print("Dataset Preview:")
print(data.head())
# Plot the data to visualize its structure
sns.scatterplot(data=data, x="Feature_1", y="Feature_2")
plt.title("Synthetic Data")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

#### Step 5: Fit a Gaussian Mixture Model (GMM)

```
# Define the GMM model
n_components = 2 # Number of Gaussian distributions (clusters)
gmm = GaussianMixture(n_components=n_components, covariance_type='full',
random_state=42)

# Fit the GMM model to the data
gmm.fit(data)

# Predict the cluster labels for each data point
labels = gmm.predict(data)

# Add the cluster labels to the dataset for visualization
data['Cluster'] = labels

# Plot the clustered data
sns.scatterplot(data=data, x="Feature_1", y="Feature_2", hue="Cluster", palette="viridis",
marker="o")

plt.title("Gaussian Mixture Model Clustering")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
```

```
plt.legend()
```

```
plt.show()
```

### **Step 6: Visualize the Gaussian Mixture Model (GMM) Components**

```
# Extract the means and covariances of the Gaussian components
```

```
means = gmm.means_
```

```
covariances = gmm.covariances_
```

```
# Plot the GMM components on top of the data
```

```
plt.figure(figsize=(8, 6))
```

```
# Plot data points
```

```
sns.scatterplot(data=data, x="Feature_1", y="Feature_2", hue="Cluster",  
palette="viridis", marker="o", s=60, alpha=0.7)
```

```
# Plot the GMM ellipses for mean, covar in zip(means, covariances):
```

```
# Plot the Gaussian components as ellipses
```

```
v, w = np.linalg.eigh(covar)
```

```
v = 2.0 * np.sqrt(2.0) * np.sqrt(v)
```

```
# Scaling factor for the ellipse
```

```
u = w[0] / np.linalg.norm(w[0])
```

```
# Normalize the eigenvector
```

```
angle = np.arctan(u[1] / u[0])
```

```
# Create the ellipse
```

```
angle = angle * 180.0 / np.pi # Convert to degrees
```

```
ellipse = plt.matplotlib.patches.Ellipse(means, v[0], v[1], angle=angle, color='red', alpha=0.3)  
plt.gca().add_patch(ellipse)
```

```
plt.title("GMM Clustering with Gaussian Components")
```

```
plt.xlabel("Feature 1")
```

```
plt.ylabel("Feature 2")
```

```
plt.legend()
```

```
plt.show()
```

### **Step 7: Model Evaluation (Optional)**

```
# Compute the log-likelihood of the data under the fitted GMM model
```

```
log_likelihood = gmm.score(data)
```

```
print(f"Log-Likelihood of the data: {log_likelihood:.2f}")
```

### **Step 8: Predict New Data Points**

```
# Example of predicting the cluster for new data points
new_data = np.array([[1.5, 2.5], [4.5, 5.5], [7.0, 8.0]])

new_labels = gmm.predict(new_data)

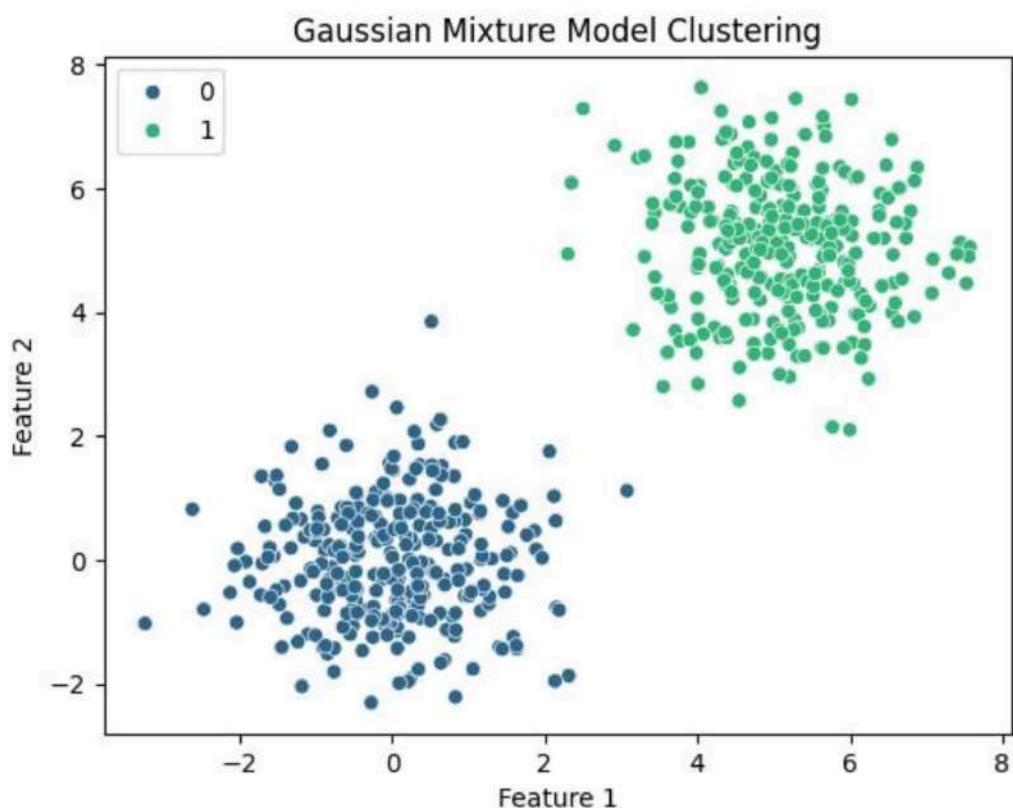
# Print the predicted clusters for the new data
# points

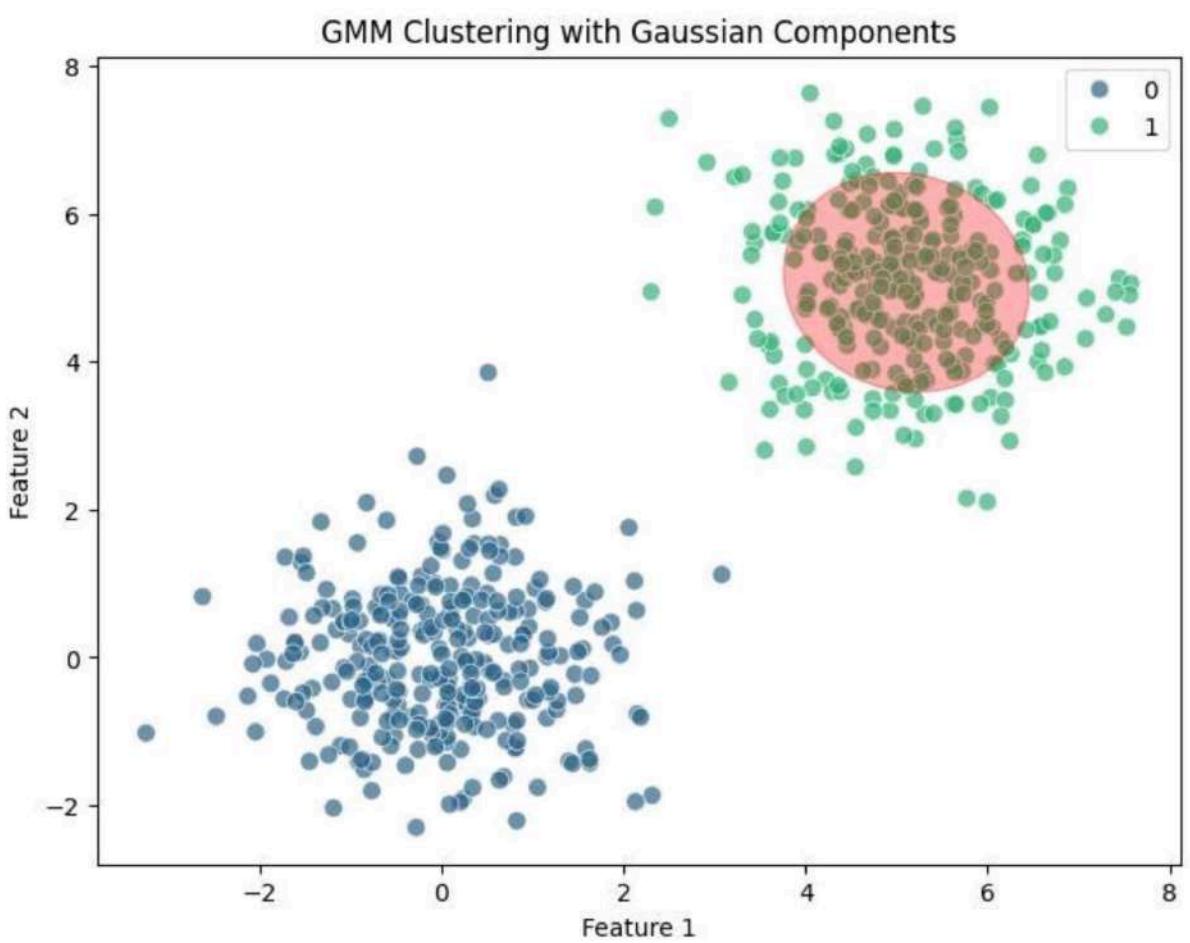
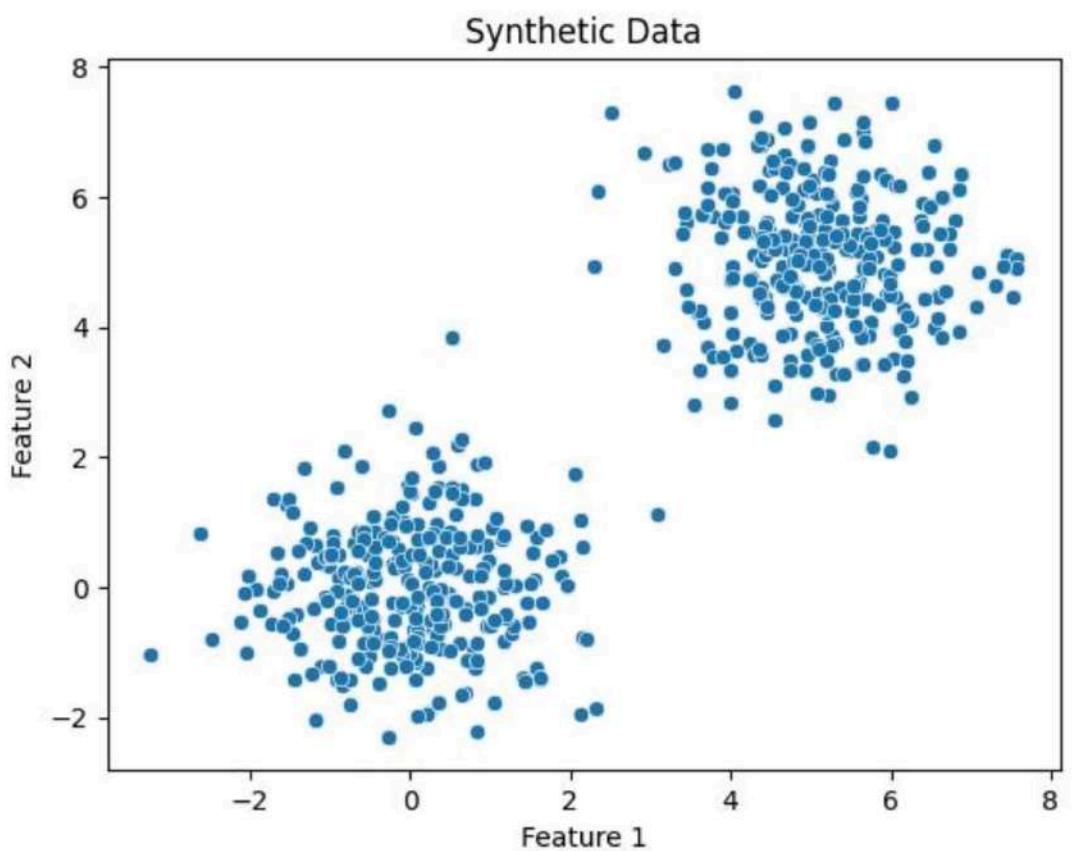
print("Predicted Clusters for New Data Points:")

for i, label in enumerate(new_labels):

    print(f'Data point {new_data[i]} is in Cluster {label}'")
```

### **Output :**





## **Practical 7 : Model Evaluation and Hyperparameter Tuning**

### **7a. Implement cross-validation techniques (k-fold, stratified, etc.) for robust model evaluation**

**Code :**

#### **1. Import Necessary Libraries**

```
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split, KFold, StratifiedKFold, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

#### **2. Generate a Synthetic Dataset**

##### **# Create a synthetic dataset with 2 classes**

```
X, y = make_classification(
    n_samples=1000, n_features=10, n_informative=8, n_redundant=2,
    n_clusters_per_class=1, random_state=42
)
```

##### **# Convert to a DataFrame for visualization**

```
df = pd.DataFrame(X, columns=[f'Feature_{i}' for i in range(1, 11)])
df['Target'] = y
```

##### **# Display the first few rows**

```
print(df.head())
```

#### **3. Split Data into Train and Test Sets**

##### **# Split data into 80% training and 20% testing**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

#### 4. Define k-Fold Cross-Validation

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
print("k-Fold Cross-Validation:")
for train_index, val_index in kf.split(X_train):
    print("TRAIN:", train_index, "VALIDATION:", val_index)
```

#### 5. Define Stratified k-Fold Cross-Validation

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
print("\nStratified k-Fold Cross-Validation:")
for train_index, val_index in skf.split(X_train, y_train):
    print("TRAIN:", train_index, "VALIDATION:", val_index)
```

#### 6. Train and Evaluate Using k-Fold Cross-Validation

##### # Initialize model

```
model = RandomForestClassifier(random_state=42)
```

##### # Perform k-Fold Cross-Validation

```
accuracies = []
```

```
for train_index, val_index in kf.split(X_train):
```

```
    X_kf_train, X_kf_val = X_train[train_index], X_train[val_index]
```

```
    y_kf_train, y_kf_val = y_train[train_index], y_train[val_index]
```

##### # Train model

```
    model.fit(X_kf_train, y_kf_train)
```

##### # Validate model

```
    y_pred = model.predict(X_kf_val)
```

```
    accuracy = accuracy_score(y_kf_val, y_pred)
```

```
    accuracies.append(accuracy)
```

```
print(f"Average Accuracy from k-Fold: {np.mean(accuracies):.2f}")
```

## 7. Hyperparameter Tuning Using GridSearchCV

```
# Define parameter grid
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
}

# Perform GridSearchCV with Stratified k-Fold
grid_search = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)

# Fit to training data
grid_search.fit(X_train, y_train)
print("Best Parameters:", grid_search.best_params_)
print("Best Cross-Validation Accuracy:", grid_search.best_score_)
```

## 8. Evaluate the Final Model

```
# Use the best model for evaluation
best_model = grid_search.best_estimator_
# Predict on test data
y_test_pred = best_model.predict(X_test)
# Evaluate performance
print("\nTest Accuracy:", accuracy_score(y_test, y_test_pred))
print("\nClassification Report:\n", classification_report(y_test, y_test_pred))
```

## # Confusion matrix

```
conf_matrix = confusion_matrix(y_test, y_test_pred)

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0', 'Class 1'])

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

## Output :

```
[[ 0 -3.358483  3.159918  0.827163  0.069658 -6.715639 -2.708559
  1  2.071819 -4.055419 -2.615948 -2.599432  3.053752  0.366795
  2 -0.633469  0.712482  2.024390 -0.432639 -1.307929  0.419320
  3 -0.464478  0.892442  2.521018  2.766580 -1.933734 -1.418018
  4  1.842426 -1.192605 -2.071386 -0.131231  0.545377  0.379060

  Feature_7  Feature_8  Feature_9  Feature_10 Target
0  0.183206  1.113502  1.730759  1.228394   1
1 -0.392171 -1.191720 -1.220516  1.899925   0
2 -1.469518 -0.719051  1.155005  2.018026   0
3  1.391760 -2.430279  1.308295 -0.270896   1
4 -0.062978 -1.325591  2.037936  0.115414   0

k-Fold Cross-Validation:
TRAIN: [ 0  1  3  4  5  6  8  9 11 12 13 14 15 16 17 18 19 20
21 22 24 25 26 27 28 32 34 35 36 37 38 40 41 42 43 44
45 46 47 48 50 51 52 53 55 56 57 58 59 60 61 62 64 68
69 70 71 73 74 75 79 80 82 83 85 87 88 89 90 91 92 93
94 95 98 99 100 102 103 104 105 106 107 108 111 112 113 114 115 116
117 119 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
138 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 156 157
158 159 160 161 162 163 164 165 166 167 169 170 171 172 173 175 176 177
178 179 180 181 182 183 184 185 186 187 188 189 190 191 193 194 195 196
197 198 201 202 203 205 206 207 212 213 214 215 217 218 220 221 222 223
224 225 226 227 228 229 230 232 233 234 236 237 238 239 240 241 242 243
245 246 247 248 249 251 252 253 255 256 257 258 259 261 262 263 264 267
268 269 270 271 272 273 274 276 277 278 279 280 282 283 284 285 287 288
289 290 291 292 293 295 297 298 299 300 301 303 304 305 307 308 309 310
311 312 313 315 317 318 319 320 321 322 324 325 328 329 330 331 332 334

785 708 789 796 791 792 793 784 797 799] VALIDATION: [- 2  0  7  8 10  18 23 29 30 31 33 39 49 54 63 65 66 67 72 76 77
78 81 86 87 91 92 100 101 102 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
137 138 139 140 141 142 143 144 145 146 147 148 150 151 152 153 154 155 156 157 159 160 161
162 166 167 168 170 171 172 173 174 175 176 180 183 184 185 186 187 188
189 190 191 192 194 195 197 199 200 201 202 203 204 205 206 207 208
209 210 211 214 215 216 217 218 219 221 222 224 225 226 228 229 230 231
232 233 235 236 237 238 240 241 242 243 244 245 246 249 250 251 252 253
254 255 256 257 258 260 261 262 263 265 266 267 268 269 270 271 272 273
274 275 276 277 278 279 280 281 282 283 284 286 287 288 289 293 294 295
296 297 298 301 302 303 304 305 306 307 308 310 312 313 314 315 316 317
318 320 321 322 323 324 325 326 327 330 333 335 336 337 338 339 340 341

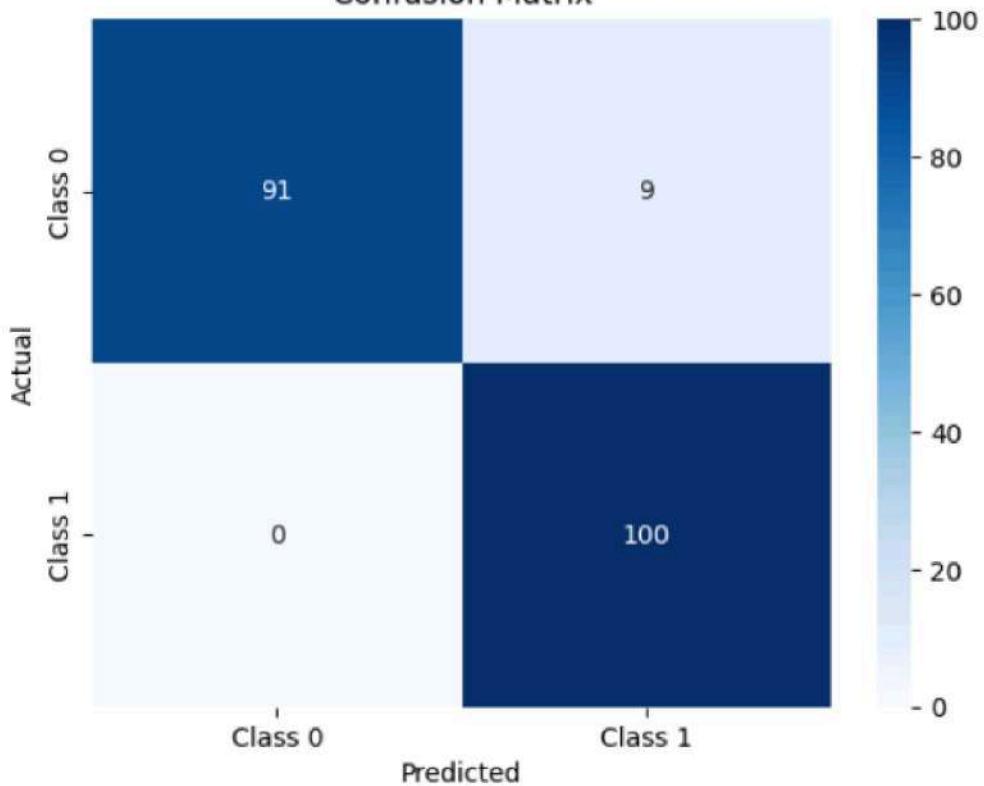
450 455 459 468 463 470 472 475 486 481 483 494 496 502 503 505 514 519
521 523 529 533 553 555 560 563 565 579 585 590 594 600 603 620 630 631
634 635 637 638 644 646 648 649 651 673 675 686 691 693 708 709 715 720
729 730 738 747 753 759 767 771 774 776 777 780 782 783 784 796
Average Accuracy from k-Fold: 0.95
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best Parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 50}
Best Cross-Validation Accuracy: 0.96

Test Accuracy: 0.95

Classification Report:
precision    recall    f1-score   support
0          1.00      0.91      0.95      100
1          0.92      1.00      0.96      100

accuracy          0.95      200
macro avg       0.96      0.96      0.95      200
weighted avg    0.96      0.95      0.95      200
```

Confusion Matrix



## **7b. Systematically explore combinations of hyperparameters to optimize model performance.(use grid and randomized search)**

**Code :**

### **1. Import Necessary Libraries**

```
import numpy as np  
import pandas as pd  
from sklearn.datasets import make_classification  
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV, StratifiedKFold  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix  
import matplotlib.pyplot as plt  
import seaborn as sns
```

### **2. Generate a Synthetic Dataset**

**# Generate a binary classification dataset**

```
X, y = make_classification(  
    n_samples=1000, n_features=12, n_informative=8, n_redundant=2,  
    n_clusters_per_class=1, flip_y=0.03, random_state=42  
)
```

**# Convert to a DataFrame for visualization**

```
df = pd.DataFrame(X, columns=[f'Feature_{i}' for i in range(1, 13)])  
df['Target'] = y
```

**# Display the first few rows**

```
print(df.head())
```

### **3. Split Data into Train and Test Sets**

**# Split data into training and testing sets**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

#### **4. Define the Model**

```
# Initialize a Random Forest classifier
```

```
model = RandomForestClassifier(random_state=42)
```

#### **5. Hyperparameter Tuning Using Grid Search**

```
# Define a parameter grid for Grid Search
```

```
param_grid = {
```

```
    'n_estimators': [50, 100, 200],
```

```
    'max_depth': [None, 10, 20],
```

```
    'min_samples_split': [2, 5, 10],
```

```
    'min_samples_leaf': [1, 2, 4]
```

```
}
```

```
# GridSearchCV with 5-fold cross-validation
```

```
grid_search = GridSearchCV(
```

```
    estimator=model,
```

```
    param_grid=param_grid,
```

```
    scoring='accuracy',
```

```
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
```

```
    verbose=1,
```

```
    n_jobs=-1
```

```
)
```

```
# Fit the model
```

```
grid_search.fit(X_train, y_train)
```

```
# Best parameters and score from Grid Search
```

```
print("Best Parameters from Grid Search:", grid_search.best_params_)
```

```
print("Best Cross-Validation Accuracy from Grid Search:", grid_search.best_score_)
```

#### **6. Hyperparameter Tuning Using Randomized Search**

```
from scipy.stats import randint
```

```

# Define a parameter distribution for Randomized Search
param_dist = {
    'n_estimators': randint(50, 300),
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': randint(2, 15),
    'min_samples_leaf': randint(1, 10)
}

# RandomizedSearchCV with 5-fold cross-validation
random_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_dist,
    n_iter=50, # Number of random combinations to try
    scoring='accuracy',
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
    verbose=1,
    n_jobs=-1,
    random_state=42
)

# Fit the model
random_search.fit(X_train, y_train)

# Best parameters and score from Randomized Search
print("Best Parameters from Randomized Search:", random_search.best_params_)
print("Best Cross-Validation Accuracy from Randomized Search:",
      random_search.best_score_)

```

## 7. Evaluate the Best Model

```

# Select the best model from Grid Search and Randomized Search
best_model = random_search.best_estimator_ # Or use grid_search.best_estimator_
# Predict on test data
y_test_pred = best_model.predict(X_test)
# Evaluate the performance

```

```

print("\nTest Accuracy:", accuracy_score(y_test, y_test_pred))
print("\nClassification Report:\n", classification_report(y_test, y_test_pred))

# Confusion Matrix

conf_matrix = confusion_matrix(y_test, y_test_pred)

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0', 'Class 1'])

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

```

## Output :

```

Feature_1  Feature_2  Feature_3  Feature_4  Feature_5  Feature_6  \
0  0.013650  0.607473 -2.096916  2.867232  2.504360  0.784101
1  0.107199  0.105735 -3.843343  1.524052 -1.619824  0.778334
2  -1.779086 -5.219831 -0.738488  2.108084 -0.803833 -3.431122
3  -4.310656 -2.268569  1.864943 -1.246116  1.268794 -2.007664
4  -3.195179 -0.671327  3.720485  0.356661  0.819486  2.670238

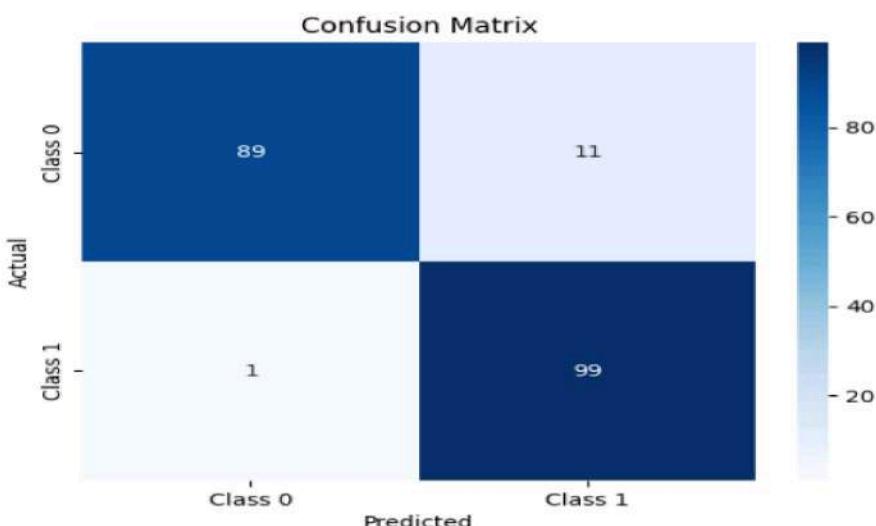
Feature_7  Feature_8  Feature_9  Feature_10  Feature_11  Feature_12  Target
0  -0.497744 -0.482072  1.112773  1.641637 -2.689832 -0.488311  1
1  0.551177 -1.843583 -0.110132 -0.494739 -0.985276 -0.978400  0
2  1.346120 -0.858351 -0.792415 -2.260815  0.238780  3.029952  0
3  -0.824133 -2.277449  0.936206  1.255903  1.386278 -0.321200  0
4  1.857477 -3.410944 -1.773719  0.656476  3.534189 -1.704889  0

Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best Parameters from Grid Search: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Best Cross-Validation Accuracy from Grid Search: 0.9487499999999999
Fitting 5 folds for each of 50 candidates, totalling 250 fits
Best Parameters from Randomized Search: {'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 9, 'n_estimators': 285}
Best Cross-Validation Accuracy from Randomized Search: 0.9487499999999999

Test Accuracy: 0.94

```

Classification Report:					
	precision	recall	f1-score	support	
0	0.99	0.89	0.94	100	
1	0.90	0.99	0.94	100	
accuracy			0.94	200	
macro avg	0.94	0.94	0.94	200	
weighted avg	0.94	0.94	0.94	200	



# **Practical 8 : Bayesian Learning**

## **Implement Bayesian Learning using inferences**

### **Code :**

#### **1. Import Necessary Libraries**

```
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

#### **2. Generate a Synthetic Dataset**

We create a dataset suitable for classification problems.

##### **# Generate a dataset with 2 classes**

```
X, y = make_classification(
    n_samples=1000, n_features=8, n_informative=6, n_redundant=2,
    n_classes=2, random_state=42)
```

##### **# Convert to DataFrame for visualization**

```
df = pd.DataFrame(X, columns=[f'Feature_{i}' for i in range(1, 9)])
df['Target'] = y
```

##### **# Display the first few rows**

```
print(df.head())
```

#### **3. Split the Dataset**

Divide the data into training and testing sets.

## **# Split data into 80% training and 20% testing**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

## **4. Bayesian Learning with Naive Bayes**

Here, we implement Bayesian Learning using the Gaussian Naive Bayes classifier.

### **# Initialize the Gaussian Naive Bayes model**

```
model = GaussianNB()
```

### **# Fit the model to the training data**

```
model.fit(X_train, y_train)
```

### **# Predict on the test data**

```
y_pred = model.predict(X_test)
```

## **5. Evaluate the Model**

We evaluate the model's performance using accuracy, classification report, and confusion matrix.

### **# Calculate accuracy**

```
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f'Test Accuracy: {accuracy:.2f}')
```

### **# Print classification report**

```
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

### **# Generate and plot confusion matrix**

```
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0', 'Class 1'])
```

```
plt.xlabel('Predicted')
```

```
plt.ylabel('Actual')
```

```
plt.title('Confusion Matrix')
```

```
plt.show()
```

## **6. Understanding Bayesian Inference**

In Bayesian Learning, the model predicts based on the probabilities:

- **Prior Probability ( $P(C)P(C)P(C)$ ):** The likelihood of each class based on historical data.
- **Likelihood ( $P(X|C)P(X|C)P(X|C)$ ):** The probability of the data given a class.
- **Posterior Probability ( $P(C|X)P(C|X)P(C|X)$ ):** Calculated using Bayes' theorem:  

$$P(C|X) = P(X|C) \cdot P(C) / P(X) = \frac{P(X|C)}{\sum P(X|C)}$$

### # Example: Compute posterior probabilities for the first test sample

```
sample = X_test[0].reshape(1, -1)
posterior_probs = model.predict_proba(sample)
print(f"Sample Features: {sample}")
print(f"Posterior Probabilities: {posterior_probs}")
print(f"Predicted Class: {model.predict(sample)})")
```

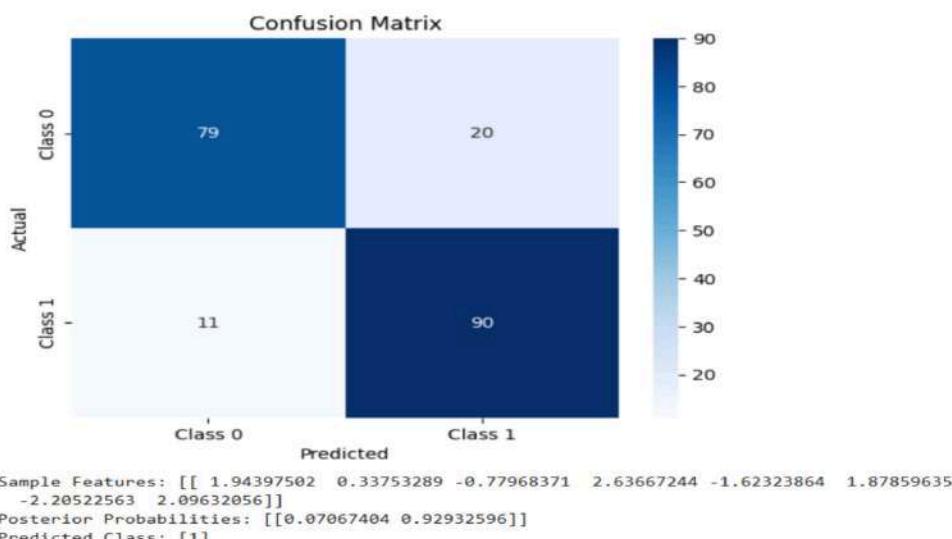
### Output :

```
Feature_1  Feature_2  Feature_3  Feature_4  Feature_5  Feature_6  \
0 -1.732538  5.260112 -2.952194 -4.603768  2.235848  1.928893
1  2.072914  2.240572 -1.385104 -2.514962 -0.984756  1.436260
2 -0.263106  1.527781 -1.872414 -0.028009  1.612809  3.264194
3 -0.164349 -0.550131 -0.019503 -0.765000  2.273523  2.084217
4 -1.419423  1.015324 -0.864441 -0.009297  0.385404  0.449093

Feature_7  Feature_8  Target
0 -0.101845  3.193487  0
1 -1.255271  2.089872  0
2 -1.296421  1.537870  0
3 -0.321931  0.426253  0
4 -0.029007 -1.982917  1
Test Accuracy: 0.84

Classification Report:
precision    recall   f1-score   support
          0       0.88      0.80      0.84      99
          1       0.82      0.89      0.85     101

accuracy           0.84      200
macro avg       0.85      0.84      0.84      200
weighted avg     0.85      0.84      0.84      200
```





NURTURING POTENTIAL

SAKET GYANPEETH'S  
**SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE**  
(Permanently Affiliated to University of Mumbai)

NAAC Accredited

Saket Vidyanagri Marg, Chinchpada Road, Katemanivali,  
Kalyan (East) -421306(Mah)

**Department of Information Technology**

This is to certify that

Mr./Ms. \_\_\_\_\_ Seat No. \_\_\_\_\_

of

**M.Sc. Information Technology**

**Part II NEP 2020 Semester III**

has satisfactorily carried out the required practical in the subject  
of \_\_\_\_\_

For the Academic year 2024 – 2025

---

**Practical In-Charge**

---

**Head of the Department**

---

**External Examiner**

**College Seal**



NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE

KALYAN (EAST)

ACADEMIC YEAR 2024-25

**M.Sc. Information Technology**

**Part II NEP 2020 Semester III**

**SUBMITTED BY**

---

**AS PRESCRIBED BY**

**UNIVERSITY OF MUMBAI**



**MUMBAI UNIVERSITY**

Prac No	Practical Description	Signature
1	<b>Deploying and Configuring Virtual Machines</b> <ol style="list-style-type: none"> <li>1. Access Your Student Desktop</li> <li>2. Create a Virtual Machine</li> <li>3. Install VMware Tools</li> <li>4. Copy Files to the Desktop</li> </ol>	
2	<b>Working with vCenter Server Appliance</b> <ol style="list-style-type: none"> <li>1. Access your vCenter Server Appliance and Configure Licenses</li> <li>2. Configure Single Sign-On and Create a Data Center Object</li> <li>3. Add Your ESXi Hosts to the vCenter Server Inventory</li> <li>4. Configure the ESXi Hosts as NTP Clients</li> <li>5. Create a Host and Cluster Folder.</li> <li>6. Create Virtual Machine and Template Folders.</li> <li>7. Navigate vSphere Client</li> </ol>	
3	<b>Users, Groups, and Permissions</b> <ol style="list-style-type: none"> <li>1. Join the vCenter Server Appliance to vclass.local Domain</li> <li>2. Add vclass.local as an Identity Source .</li> <li>3. View Active Directory Users.</li> <li>4. Assign Object Permissions to an Active Directory User</li> <li>5. Assign Root-Level Global Permission</li> <li>6. Log In with Windows Session Authentication</li> <li>7. Use an Active Directory User to Manage a Virtual Machine</li> </ol>	
4	<b>Using Standard Switches</b> <ol style="list-style-type: none"> <li>1. View the Standard Switch Configuration</li> <li>2. Create a Standard Switch with a Virtual Machine Port Group.</li> <li>3. Attach Your Virtual Machines to the New Virtual Machine Port Group</li> </ol>	

5	<b>Accessing iSCSI Storage</b> <ol style="list-style-type: none"> <li>1. Validate an Existing ESXi Host iSCSI Configuration</li> <li>2. Add a VMkernel Port Group to a Standard Switch</li> <li>3. Configure the iSCSI Software Adapter Connect the iSCSI Software Adapters to Storage</li> </ol>	
6	<b>Managing VMFS Datastores</b> <ol style="list-style-type: none"> <li>1. Create VMFS Datastores for the ESXi Host</li> <li>2. Expand a VMFS Datastore to Consume Unused Space on a LUN.</li> <li>3. Remove a VMFS Datastore.</li> <li>4. Extend a VMFS Datastore</li> <li>5. Create a Second Shared VMFS Datastore Using iSCSI</li> </ol>	
7	<b>Accessing NFS Storage</b> <ol style="list-style-type: none"> <li>1. Configure Access to NFS Datastores</li> <li>2. View NFS Storage Information</li> </ol>	
8	<b>Using Templates and Clones</b> <ol style="list-style-type: none"> <li>1. Create a Virtual Machine Template</li> <li>2. Create Customization Specifications</li> <li>3. Deploy a Virtual Machine from a Template</li> </ol>	
9	<b>Modifying Virtual Machines</b> <ol style="list-style-type: none"> <li>1. Clone a Powered-On Virtual Machine</li> <li>2. Increase the Size of a VMDK File</li> <li>3. Adjust Memory Allocation on a Virtual Machine.</li> <li>4. Rename a Virtual Machine in the vCenter Server Appliance Inventory</li> <li>5. Add and Remove a Raw LUN on a Virtual Machine</li> </ol>	
10	<b>Migrating Virtual Machines .</b> Migrate Virtual Machine Files from Local Storage to Shared Storage <ol style="list-style-type: none"> <li>1. Create a Virtual Switch and a VMkernel Port Group for vSphere vMotion Migration .</li> <li>2. Prepare Virtual Machines to Demonstrate vSphere vMotion Migration</li> <li>3. Perform vSphere vMotion Migrations of Virtual Machines</li> <li>4. Perform Compute Resource and Storage Migrations</li> </ol>	

11	<b>Managing Virtual Machines.</b> 1. Unregister a Virtual Machine from the vCenter Server Appliance Inventory 2. Register a Virtual Machine in the vCenter Server Appliance Inventory . 3. Unregister and Delete Virtual Machines from the Datastore. 4. Take Snapshots of a Virtual Machine 5. Add Files and Take Another Snapshot of a Virtual Machine. 6. Revert the Virtual Machine to a snapshot 7. Delete an Individual Snapshot 8. Delete All Snapshots.	
12	<b>Managing Resource Pools .</b> 1. Create CPU Contention . 2. Create Resource Pools . 3. Verify Resource Pool Functionality	
13	<b>Monitoring Virtual Machine Performance .</b> 1. Create the CPU Workload .. 2. Use Performance Charts to Monitor CPU 3. Undo Changes Made to the Virtual Machines	
14	<b>Using vSphere HA. . . .</b> 1. Create a Cluster Enabled for vSphere HA 2. Add Your ESXi Hosts to the Cluster 3. Test the vSphere HA Functionality 4. View the vSphere HA Cluster Resource Usage. 5. Manage vSphere HA Slot Size 6. Configure a vSphere HA Cluster with Strict Admission Control	

Module 1.1 – Access your Student Desktop

Launch Firefox

[22]



1. Click on the Firefox Icon on the Linux Task Bar

Select esx-03a

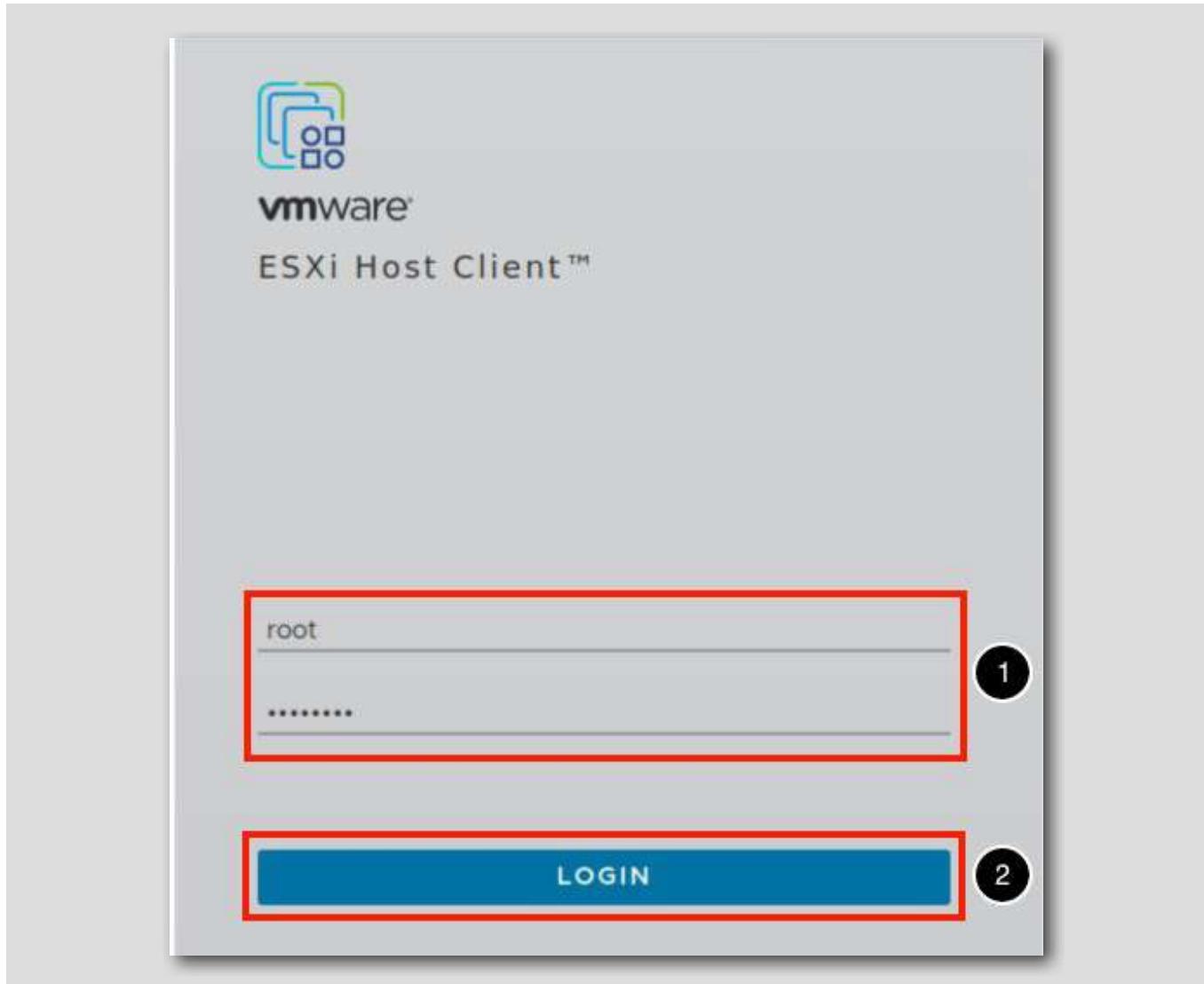
[23]



1. From the Bookmarks bar, select esx-03a Admin

Login

[24]



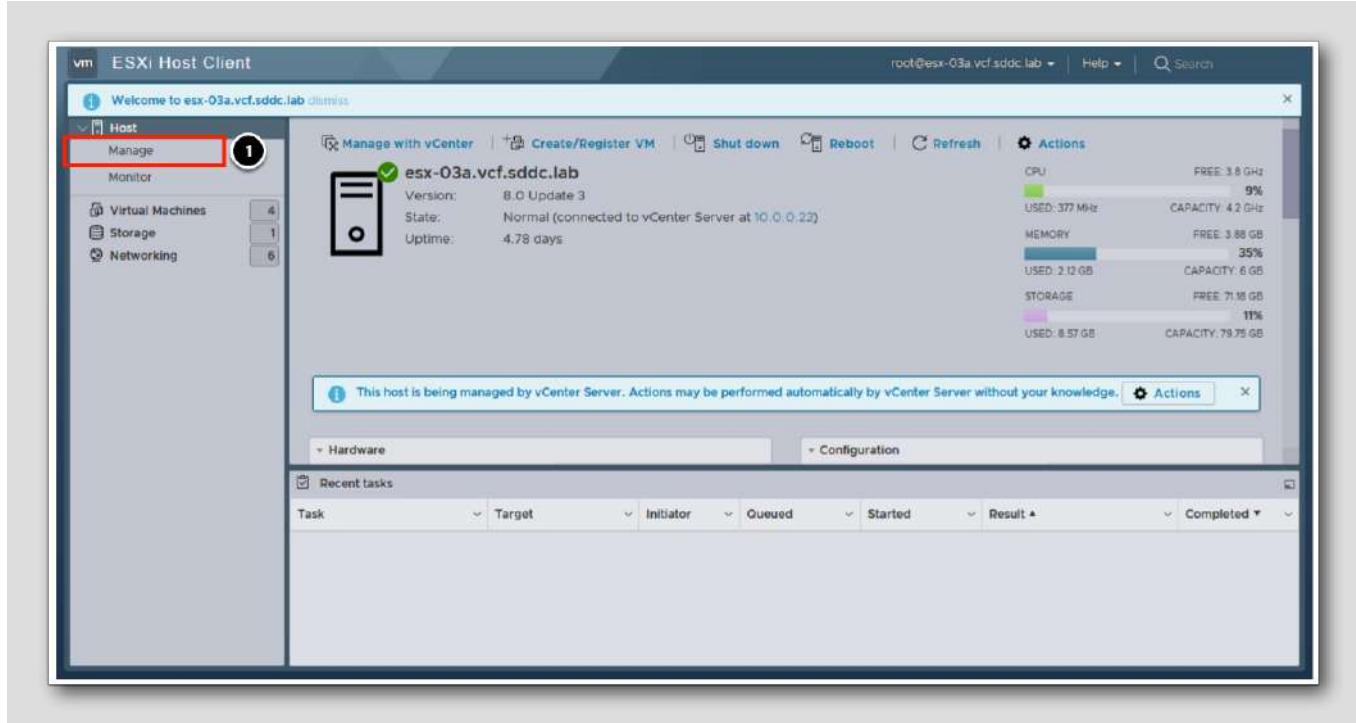
1. Login with the following credentials:

- Username: root
- Password: VMware123!

2. Click the Login button

## ESXi Host Client

[25]



The ESXi Host, in this case, esx-03a, can now be directly managed. This can be useful in test/dev environments where a vCenter Server is not present or in a production environment where the vCenter Server is not reachable.

The initial screen shows high-level details and recent tasks. There are also various power options for the host and an Actions menu for the most common tasks. Note that the server is currently in Maintenance Mode, which will be discussed in a future lesson. 1. Click on Manage

## System

[26]

Key	Name	Value	Default	Overridden
Annotations.WelcomeMessage	The welcome message in the initial ...			False
CBRC.DCacheMemReserved	Memory consumed by CBRC Data C...	400	400	False
CBRC.Enable	Enable Content Based Read Cache	false	false	False

Quick filters... 1213 items

1. Click on the System tab.

You can find that the most common options set are the date and time for the host. It can be set and synchronized with an NTP server or set manually. In addition, Autostart settings for the host can be configured here as well.

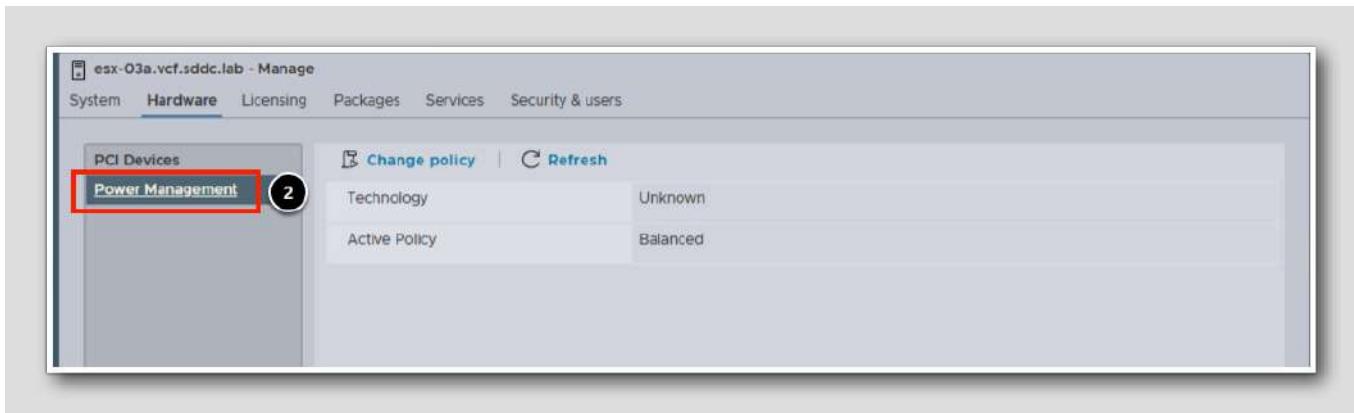
## Hardware

[27]

Address	Description	SR-IOV	Passthrough	Hardware
0000:00:0...	Intel Corporation Virtual Machine Chipset	Not capable	Not capable	
0000:00:0...	Intel Corporation 440BX/ZX/DX - 82443BY/ZK/DX AGP bri...	Not capable	Not capable	

Quick filters... 43 items

1. Click on the Hardware tab



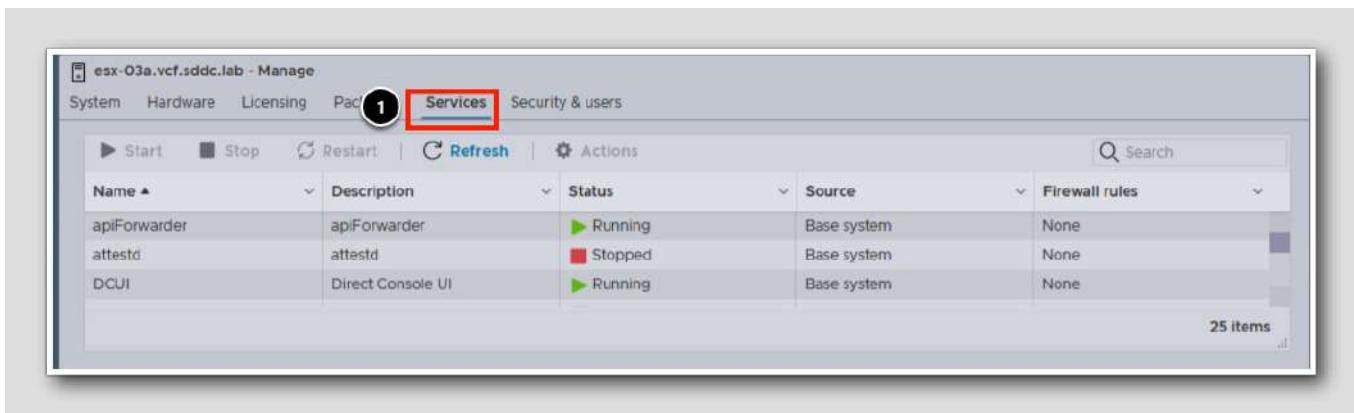
Note: Due to this being a nested virtual environment, seeing items such as "Unknown" under 'Technology' is expected behavior. You can safely ignore this and move on.

2.Click Power Management

This is where power management policies can be set for the host.

## Services

[28]

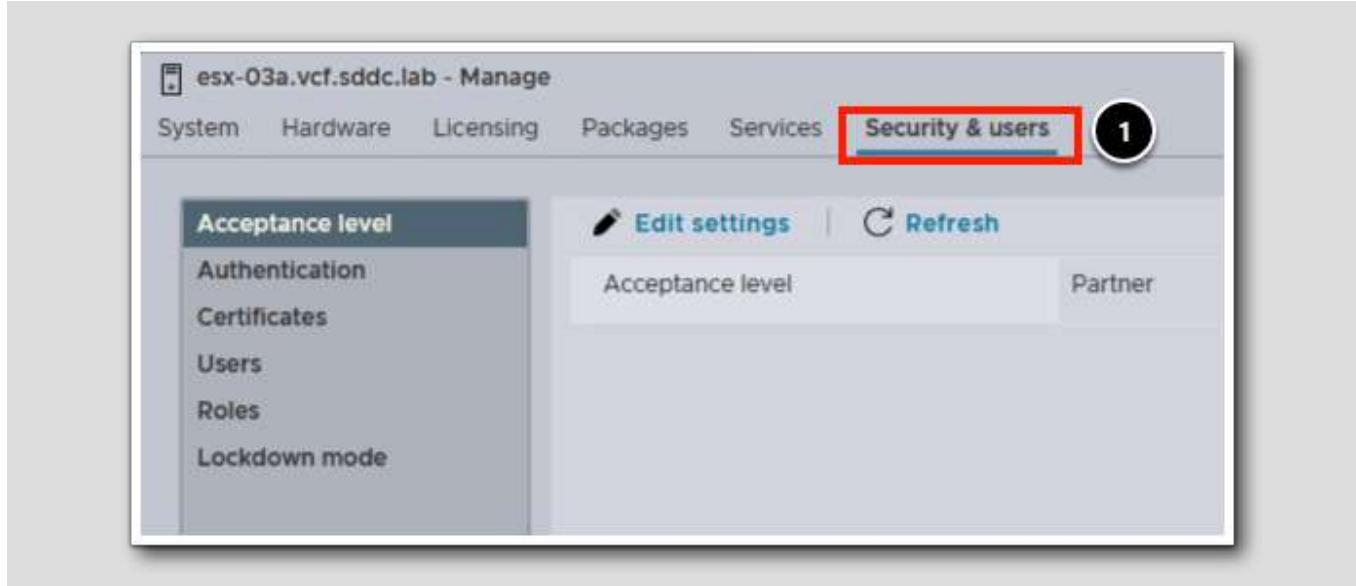


1. Click the Services tab

Services like SSH access and the Direct Console UI can be stopped and started from this screen.

## Security and Users

[29]

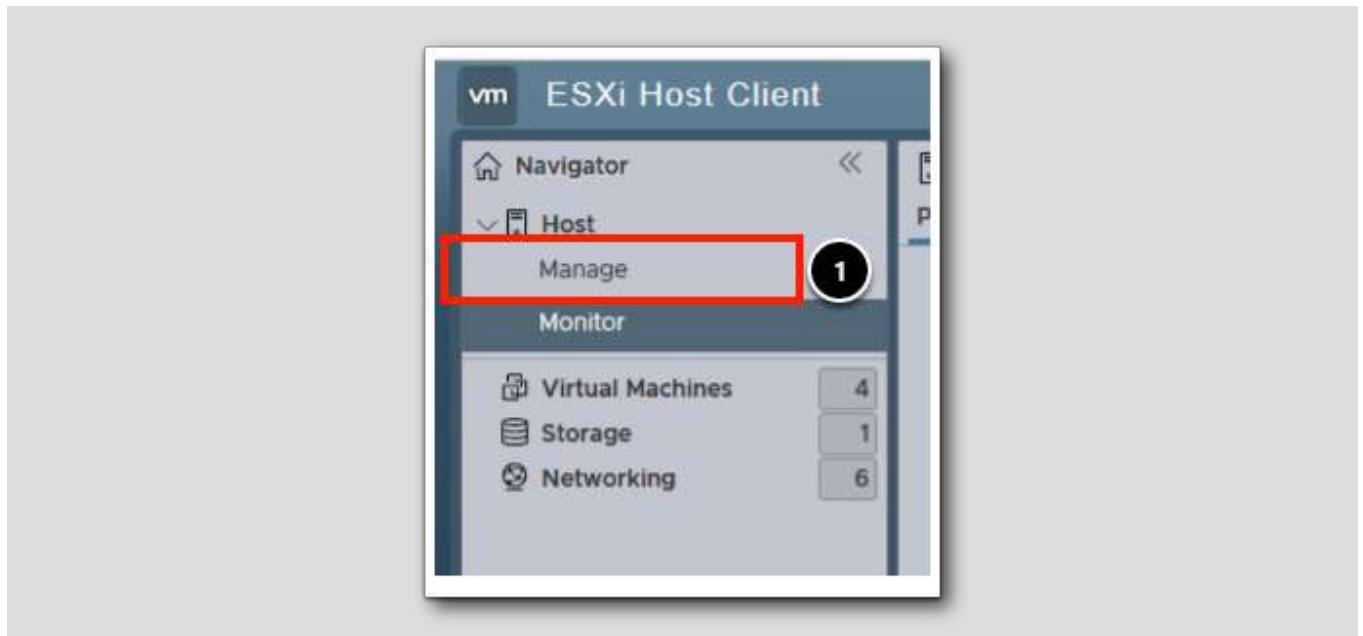


On the Security & Users tab, security options such as authentication to Active Directory and Certificates can be set here. There is also the ability to create additional roles and user accounts for the host itself. This option uses accounts that are local only to the host and not shared with any other hosts or vCenter Server. vCenter Server is set up to use single sign-on which makes account management much easier. This will be reviewed in the lessons that follow.

1. Click on Security & users

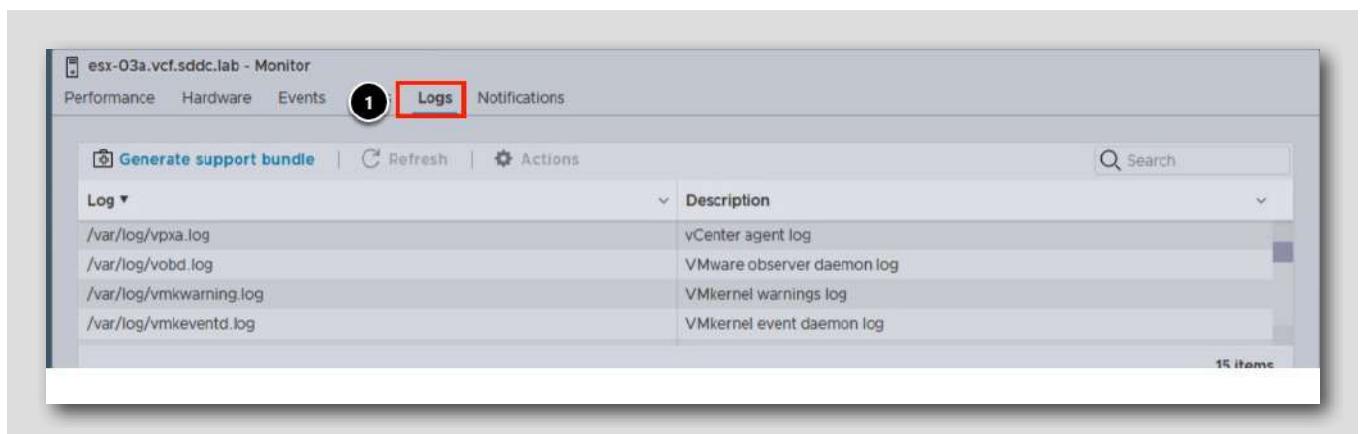
Monitor

[30]



The Monitor section includes Performance Charts, Hardware monitoring, an event log and other useful monitoring information.

1. Click on Monitor

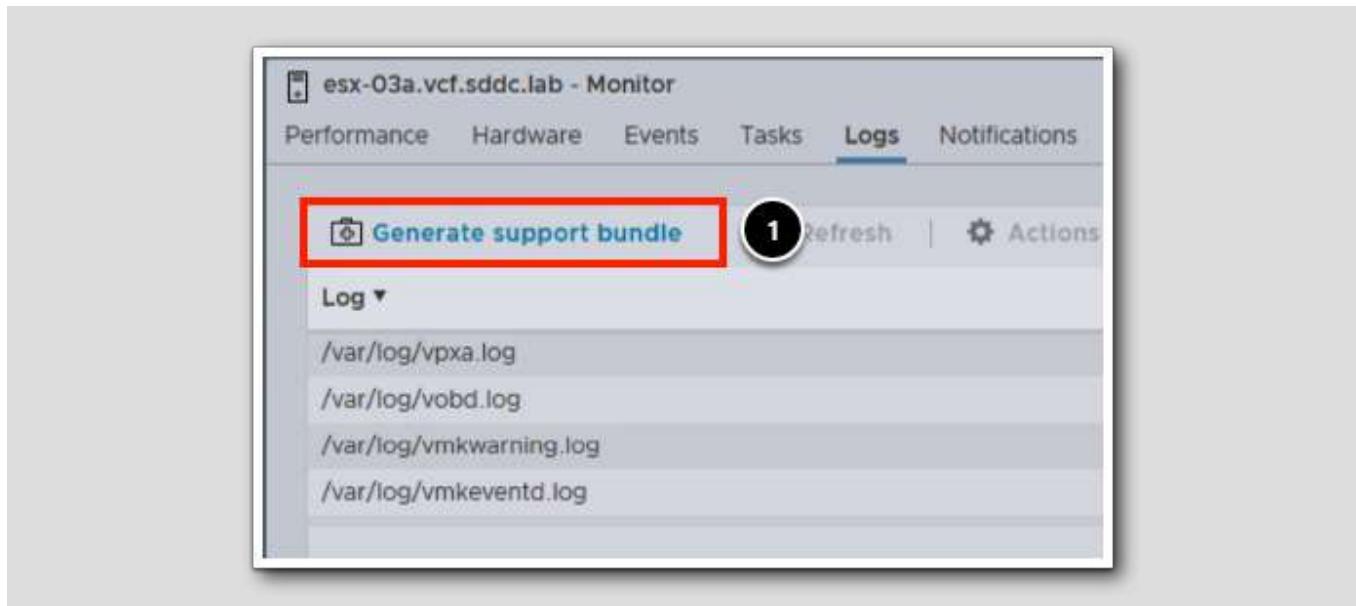


1. Click the Logs tab

On the Logs tab, a support bundle can be created that includes log files and system information that can be helpful in troubleshooting issues.

## Generate Support Bundle

[31]



1. Click the Generate support bundle button

This operation will automatically download the support file. It will take a couple of minutes.



You may be asked to provide credentials. Use the same information you used to log in:

- Username: root
- Password: VMware123!

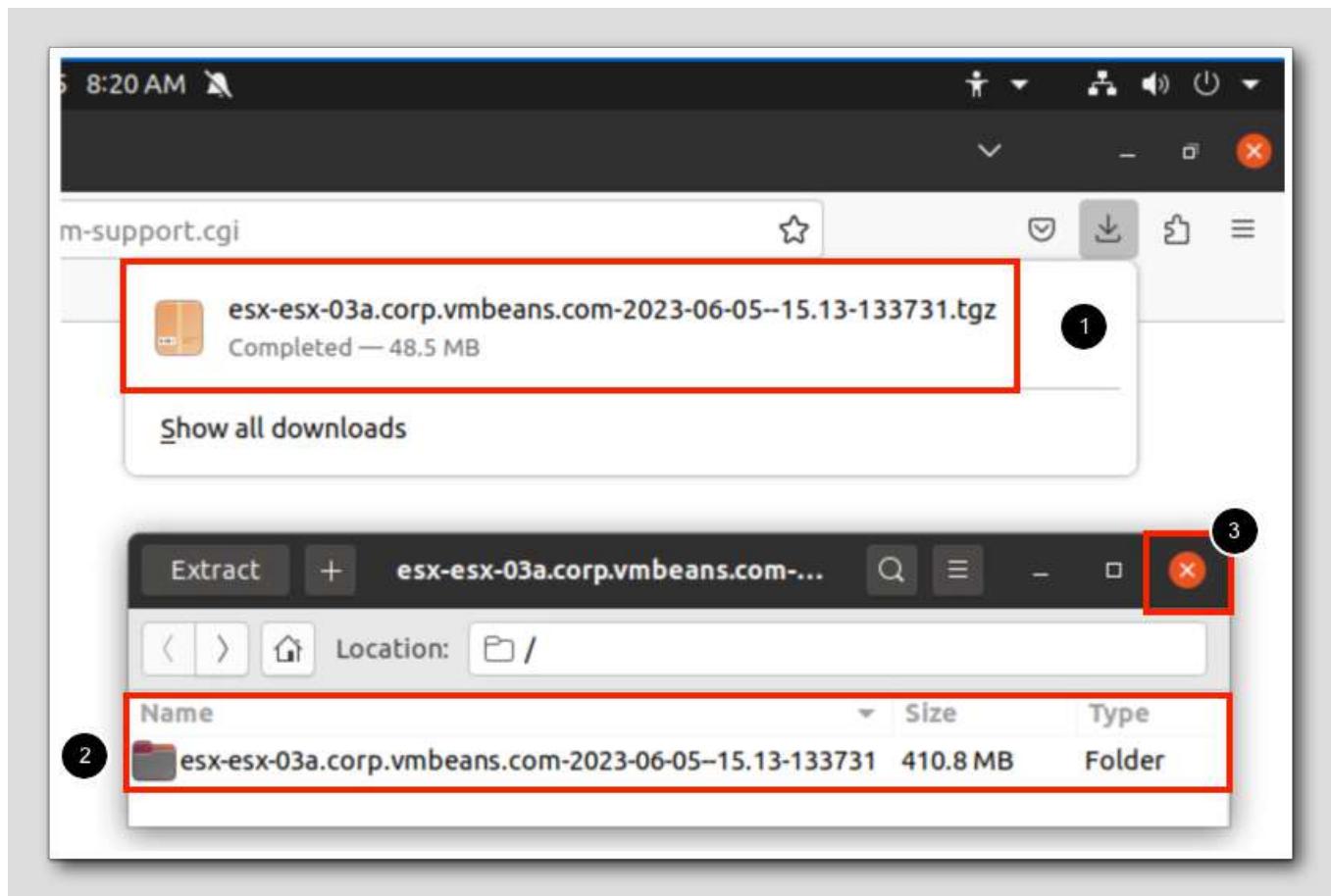
1. Click on Log in

## Save the Support Bundle

[32]



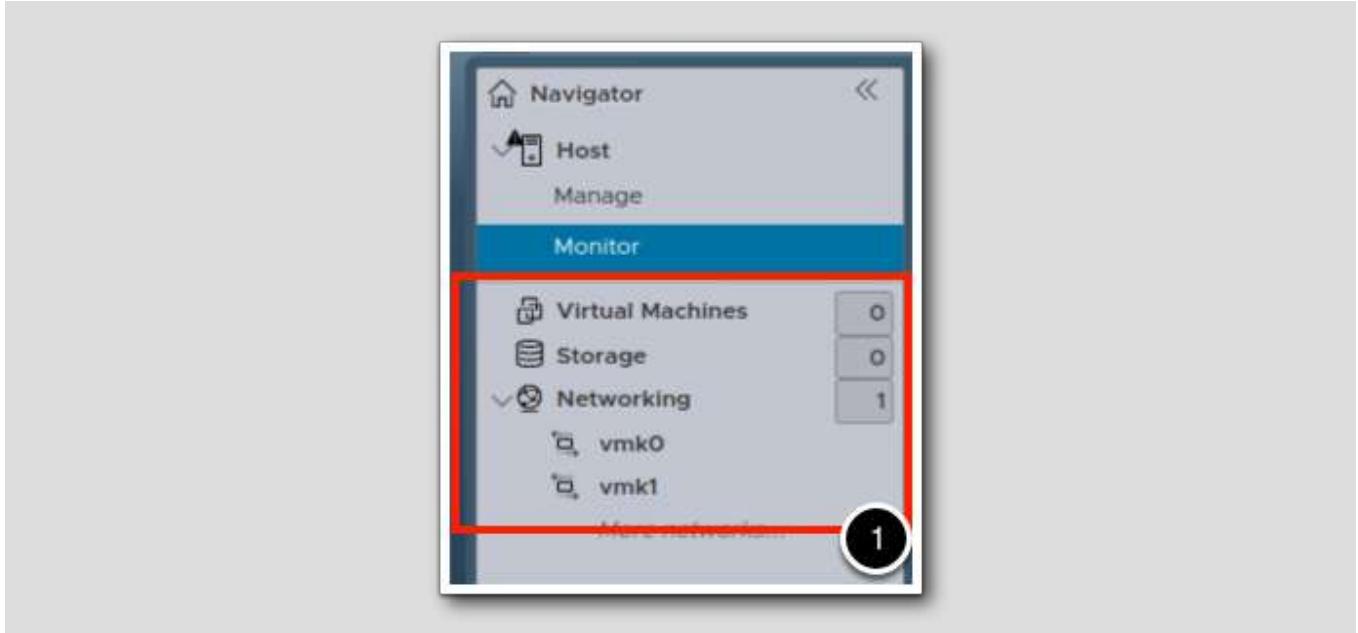
1. Click on Save if the pop up shows up.



1. Click on the downloaded file
2. A pop-up window will appear with the downloaded support file. Review file if needed.
3. Close window when finished.

## VMs, Storage and Networking

[33]



1. In addition to managing and monitoring the host, Virtual Machines can be created, Storage and Networking can be configured at the host level.

Since these features will be covered throughout the lab and the actions performed are identical, just at the vCenter Server level, we will not be reviewing them here.

The ESXi Host Client can be very useful in situations where a vCenter Server is not present to manage the host. However, when a vCenter Server is present, it is the preferred option and provides better tools to manage your infrastructure as a whole.

## Practical 1.2 – Create a Virtual Machine

Launch Firefox

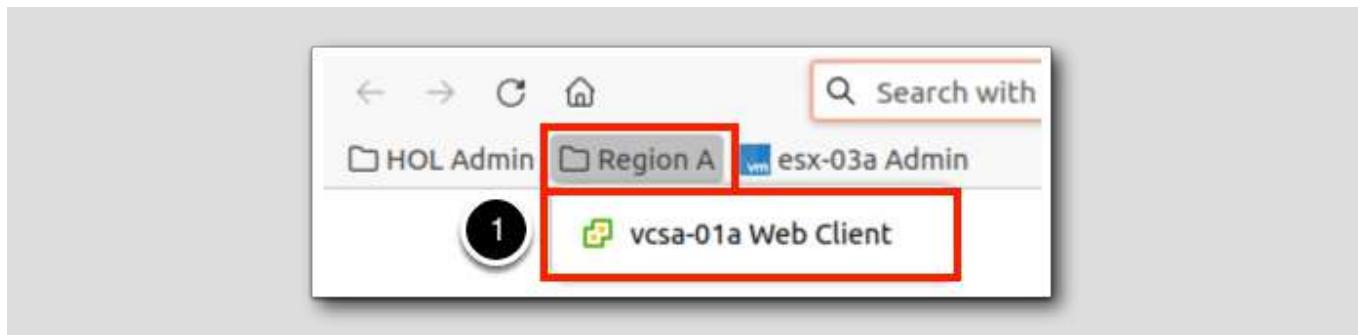
[44]



If you are not already in Firefox, click on Firefox on your desktop. If you are already in Firefox, open a new tab.

Select vSphere Client

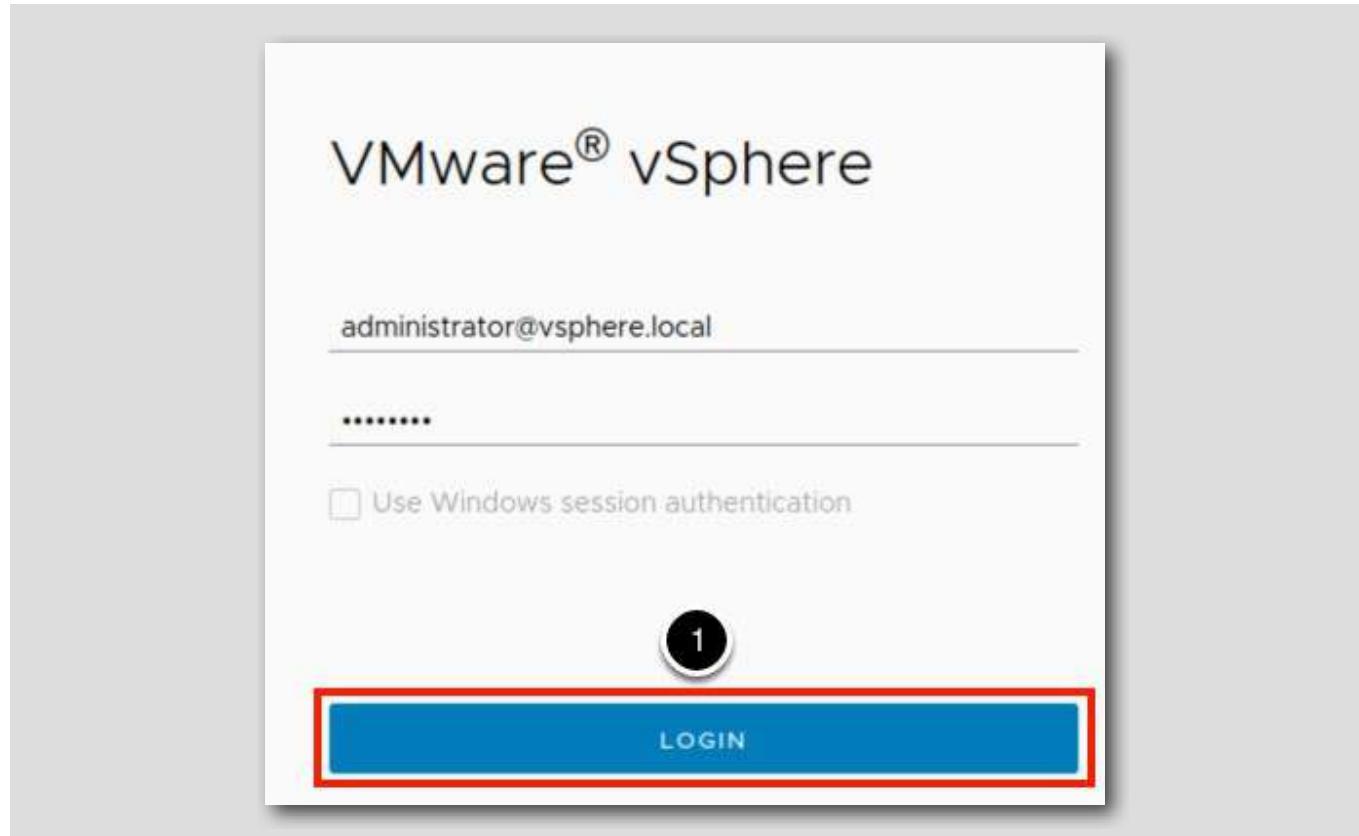
[45]



1. Under the Region A Bookmark's Folder, click the vSphere Web Client bookmark.

## Login to vCenter

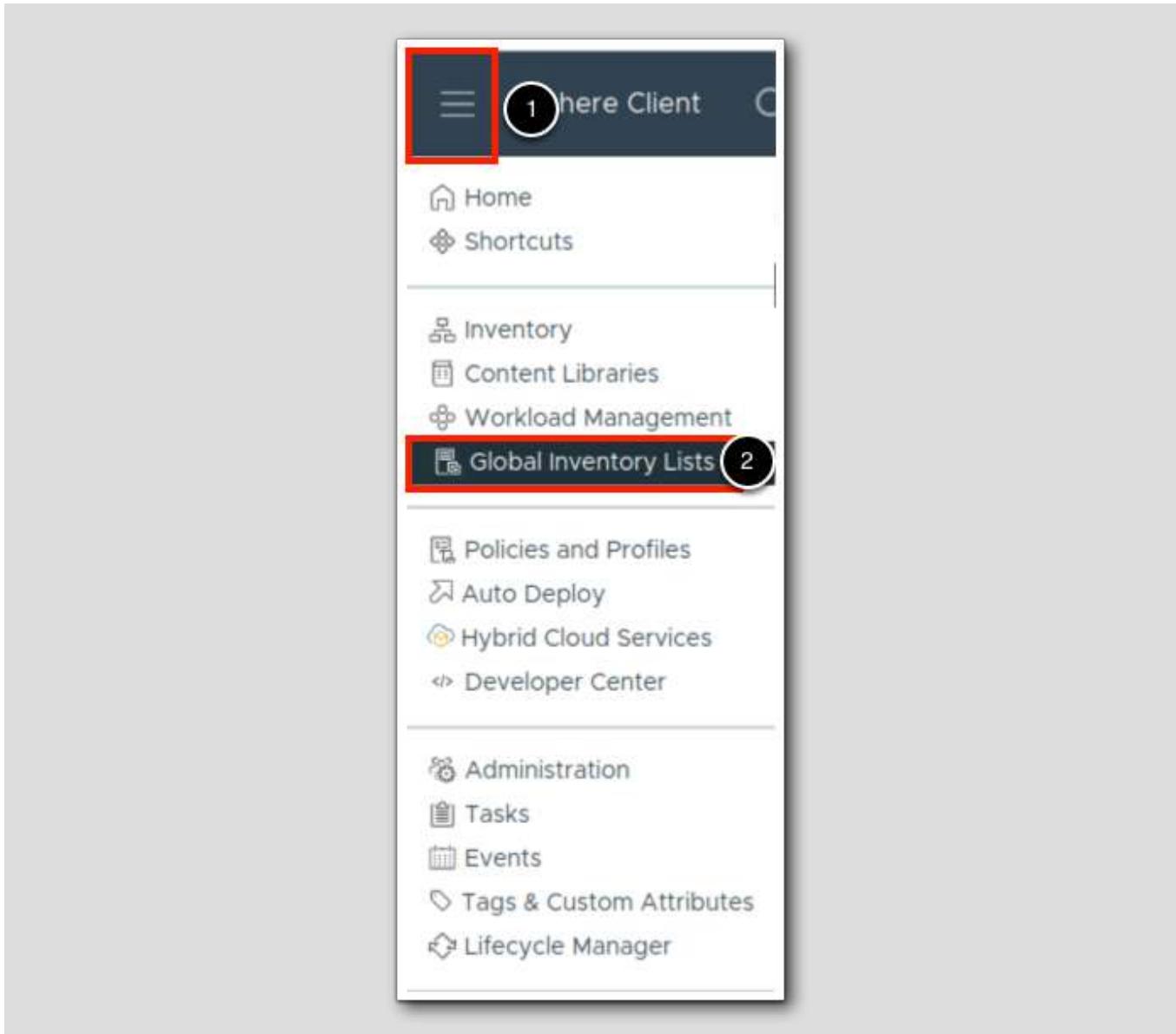
[46]



If your 'LOGIN' button is grey or cannot be click, you can click on the username and password fields, this should enable your 'LOGIN' button.

1. Click on Login to proceed - The credentials should already be pre-populated

## vCenter Inventory



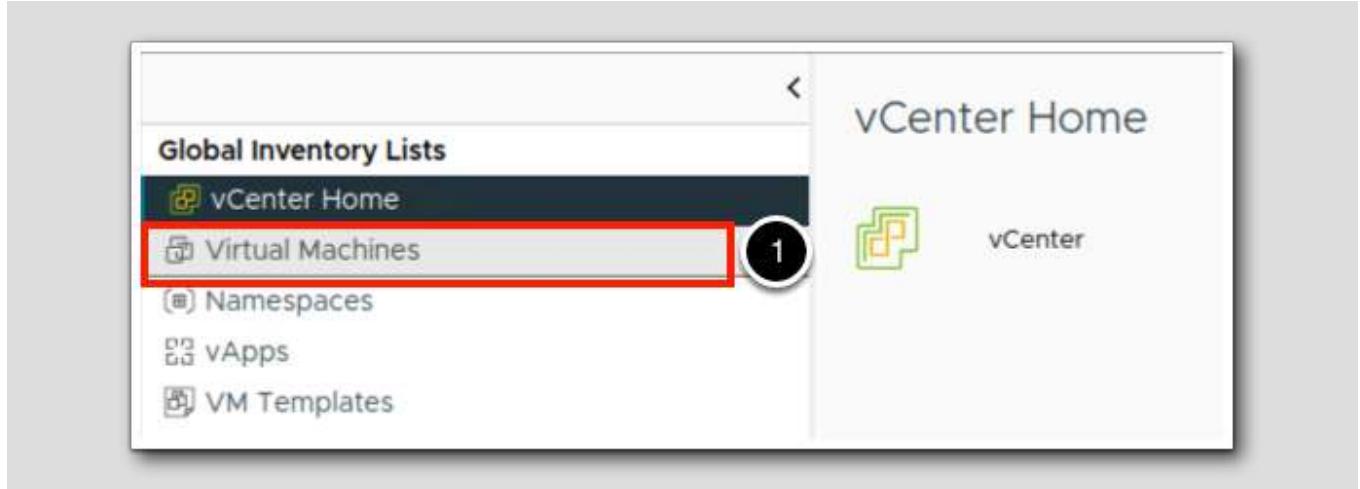
By default, you are brought to a view that shows the Hosts and Clusters attached to vCenter. Get a more complete look by viewing the Global Inventory Lists.

1. Click on the Menu hamburger icon on the top left and select Global Inventory Lists.

Clicking Global Inventory Lists will take you to the inventory page where you find all the objects associated with vCenter Server systems such as data centers, hosts, clusters, networking, storage, and virtual machines.

## Child objects, Data Centers, and Hosts

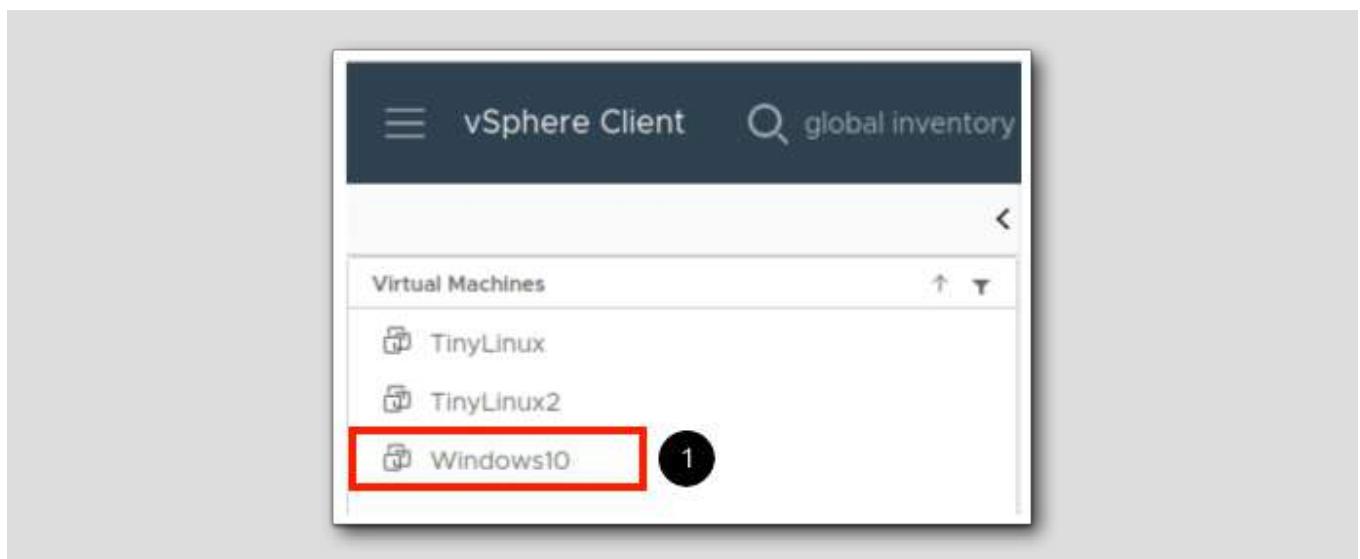
[48]



1. Click the "Virtual Machines" inventory item. By selecting this inventory item, you are presented with a list of the VMs which are located in this environment.

## Virtual Machine Summary

[49]



Here are all the virtual machines associated with this vCenter instance.

1. Click the "Windows10" virtual machine.

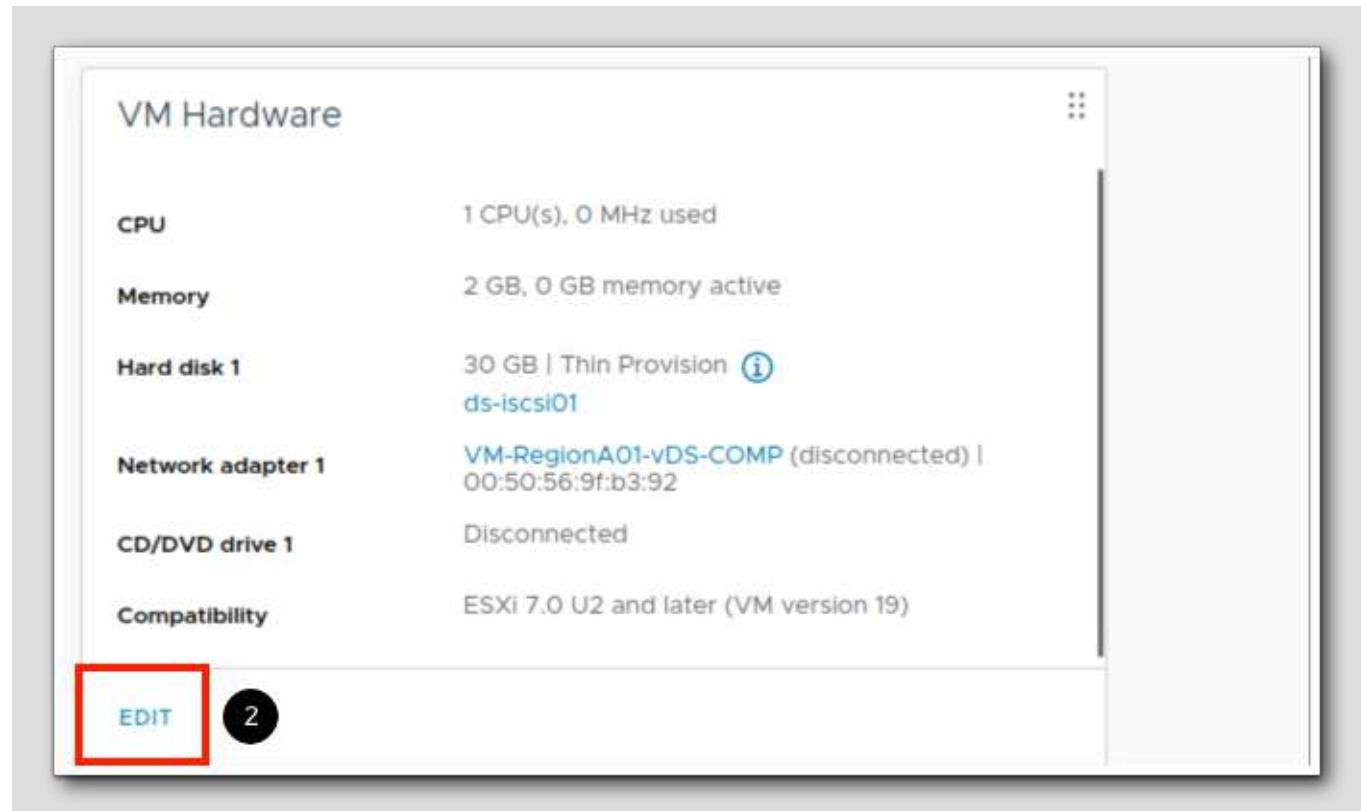
The screenshot shows the VMware vSphere Web Client interface. The top navigation bar includes tabs for 'Windows10' (selected), 'Editor', 'Configure', 'Permissions', 'Datastores', 'Networks', 'Snapshots', and 'Updates'. The 'Summary' tab is highlighted with a red box. The main content area is divided into several sections:

- Guest OS:** Shows 'Powered Off' status with 'LAUNCH REMOTE CONSOLE' and 'LAUNCH WEB CONSOLE' buttons.
- Virtual Machine Details:** Includes:
  - Power Status:** Powered Off
  - Guest OS:** Microsoft Windows 10 (64-bit)
  - VMware Tools:** Not running, version: 12.294 (Current)
  - DNS Name:** Windows10
  - IP Addresses:** Not assigned
  - Encryption:** Not encrypted
- Capacity and Usage:** Last updated at 9:26 AM.
  - CPU:** 0 MHz used, 1 CPU allocated
  - Memory:** 0 MB used, 2 GB allocated
  - Storage:** 17.77 GB used, 32.23 GB allocated
- VM Hardware:** Includes:
  - CPU:** 1 CPU(s), 0 MHz used
  - Memory:** 2 GB, 0 GB memory active
  - Hard disk 1:** 30 GB | Thin Provision
- PCI Devices:** (empty)
- Related Objects:** Cluster (RegionA01-COMPO), Host (esx-D2a.corp.vmbeans.com), Networks
- Tags:** (empty)

1. You will land in the "Summary" Tab for that virtual machine. In this page, you are able to see all the details regarding the virtual machine.

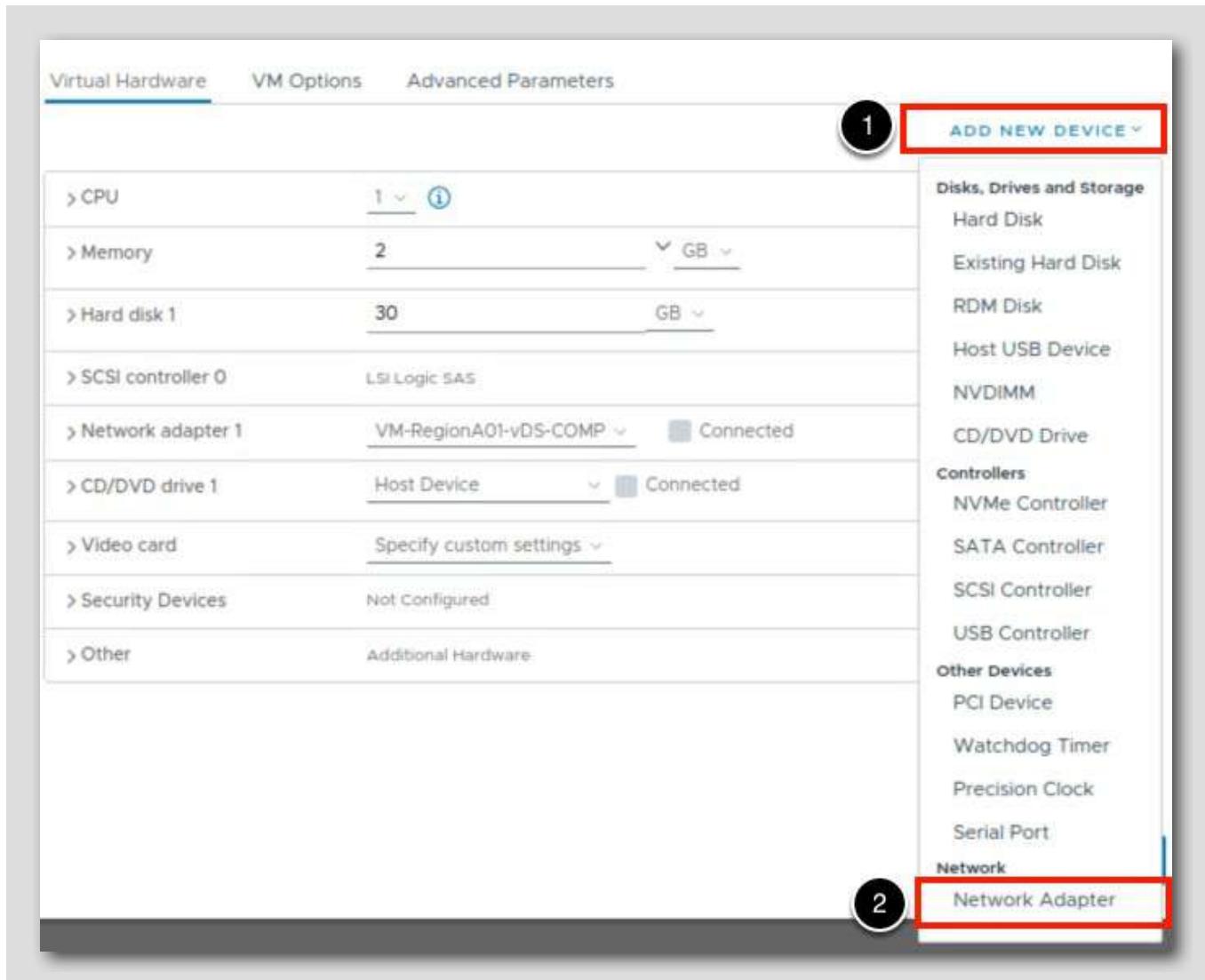
## Edit the Settings for the Virtual Machine

[50]



1. Review the VM Hardware details for the windows10 virtual machine. Note that there is currently only one network adapter.
2. Click on Edit so a second network adapter can be added to the virtual machine. You may need to scroll down to see this option.

## Add a Second Network Adapter

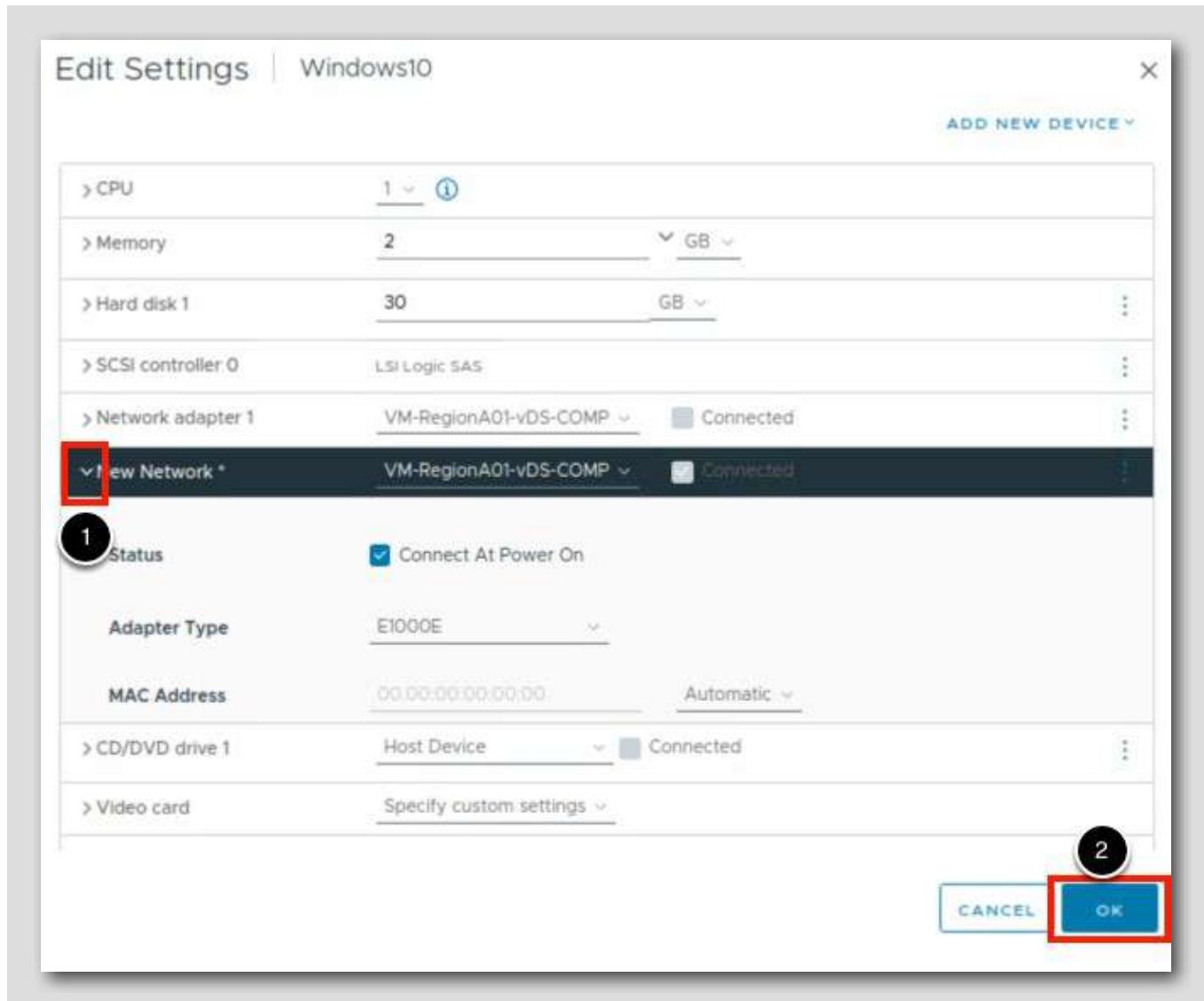


Add another network adapter to the windows10 machine.

1. In the Edit settings window, click the Add New Device drop-down menu.
2. Select Network Adapter from the list. You may need to scroll down to see the option.

## Configure the Second Network Card

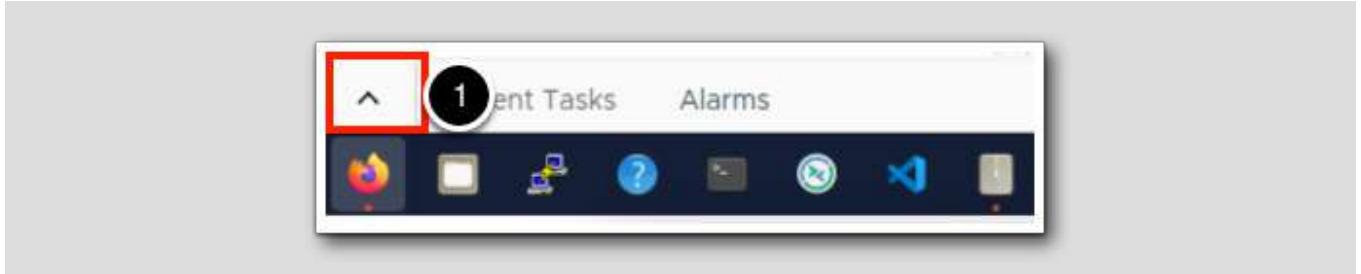
[52]



1. Click the arrow next to the New Network to expand and view its settings. Notice that the MAC address is blank at this point. A new MAC address will be generated once this NIC is added or we are able to specify (with some rules) our own MAC address.
2. Click on OK to add the device to the VM. When you select OK a new task will be created.

## Recent Tasks List (1/2)

[53]



1. Click on the Arrow to view the Recent Tasks to watch the task's progress.

## Recent Tasks List (2/2)

[54]

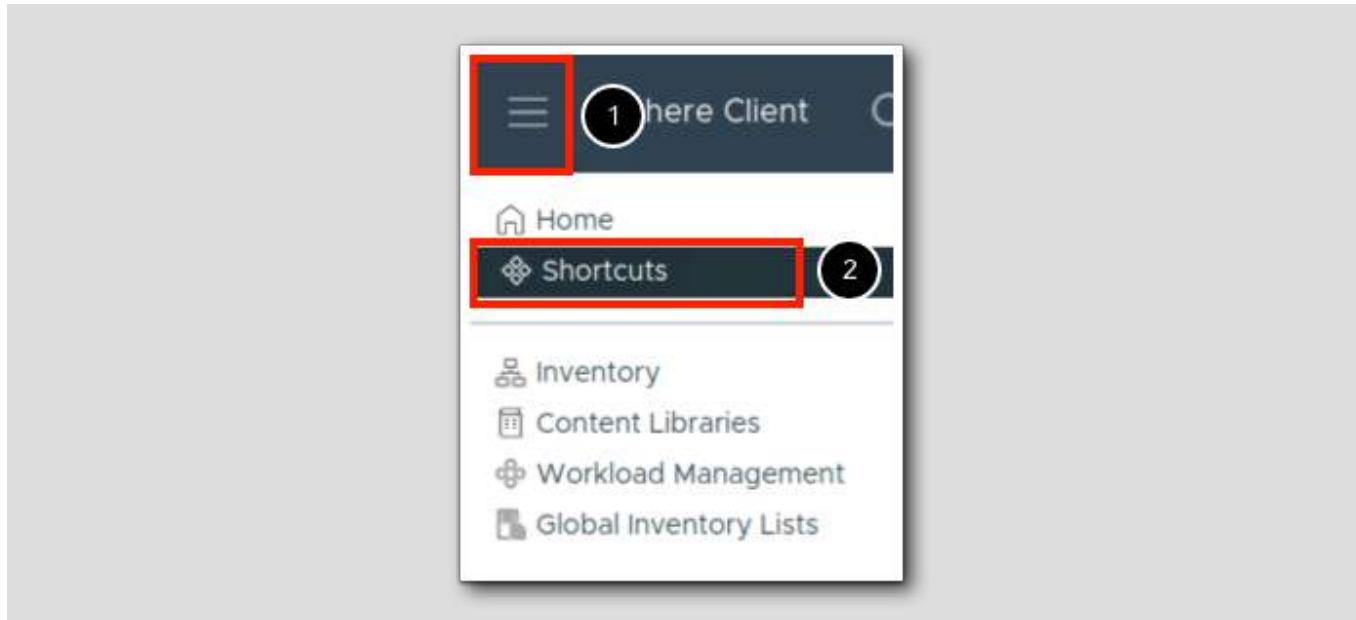
Recent Tasks		Alarms	
Task Name	Target	Status	Details
Reconfigure virtual mach...	Windows10	Completed	VSPHERE.LOCAL\Administrator 5 ms
<a href="#">All</a> <a href="#">More Tasks</a>			

Review the "Recent Tasks" list. Once the task is complete, a second Network Adapter should be shown in the "VM Hardware" section. Note the networks are in a disconnected state because the VM is powered off.

Once you are done viewing the Recent Tasks list, click the down-arrows to minimize it.

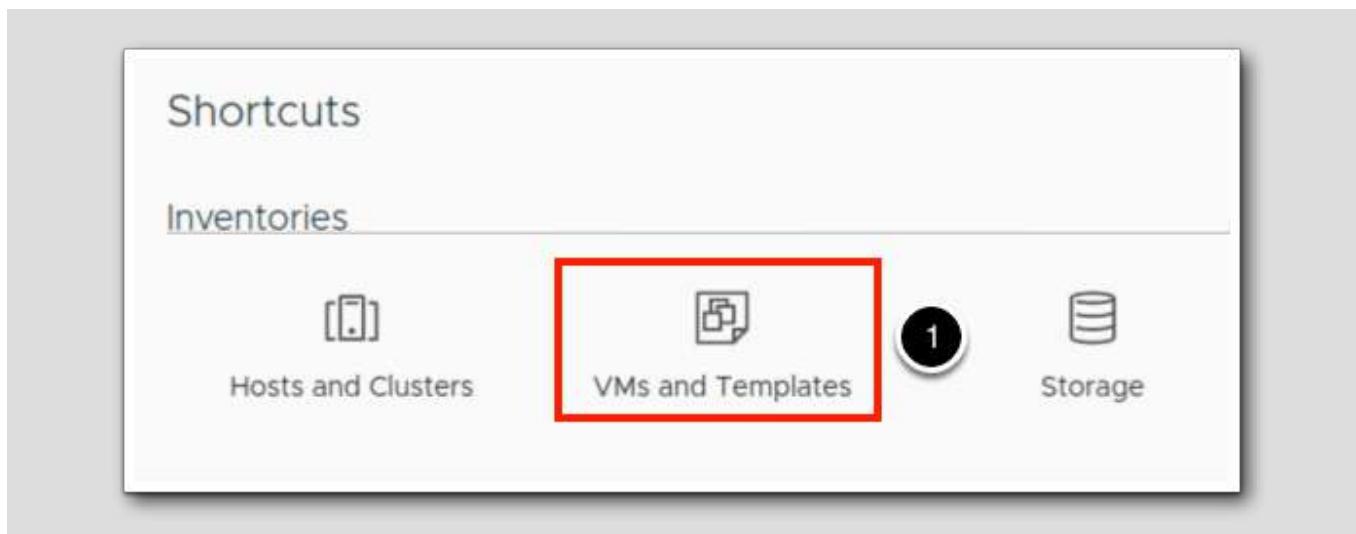
## Create a Virtual Machine

[55]



In the next steps, we will create a virtual machine and then, install an operating system.

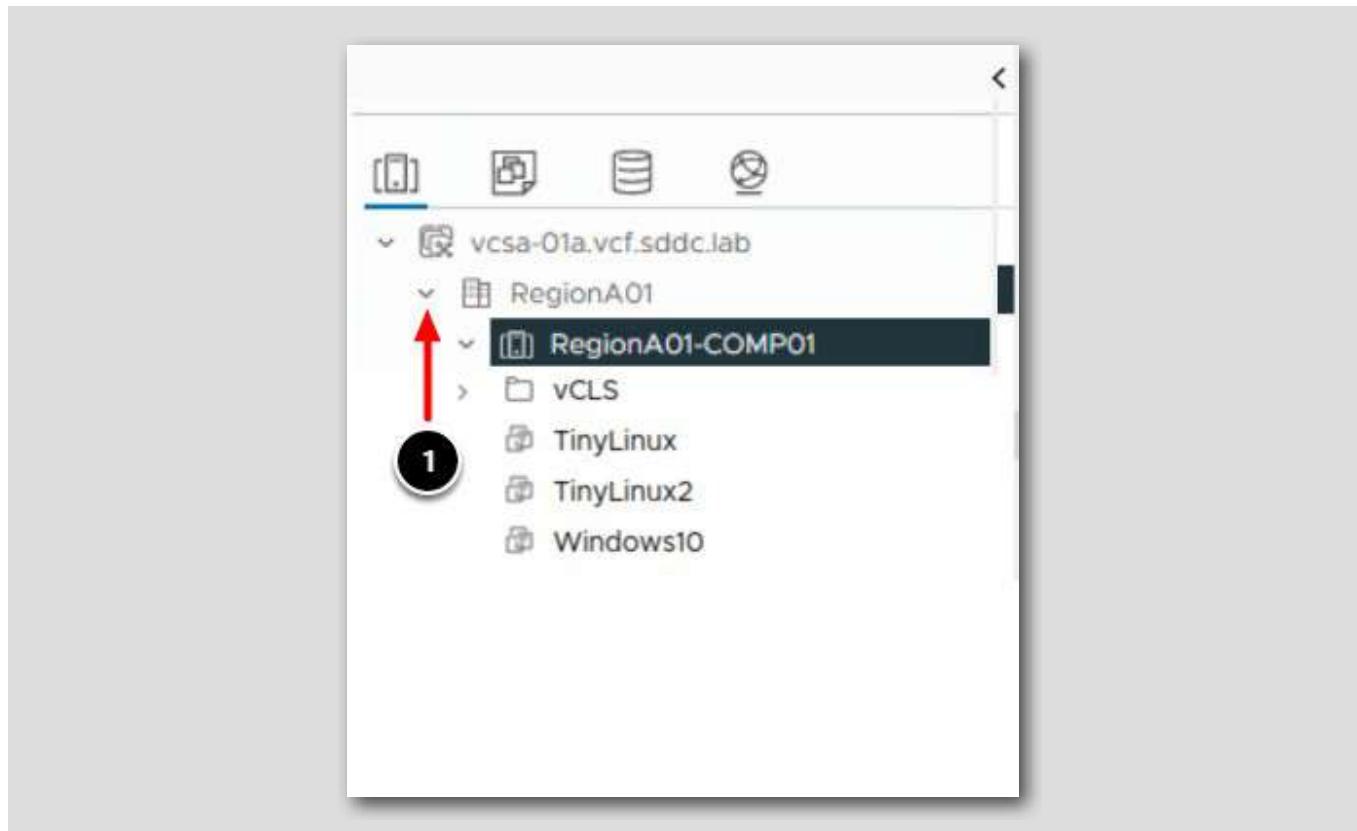
1. To return to the VMs and Templates view, click on Menu.
2. Click into Shortcuts



1. Select VMs and Templates.

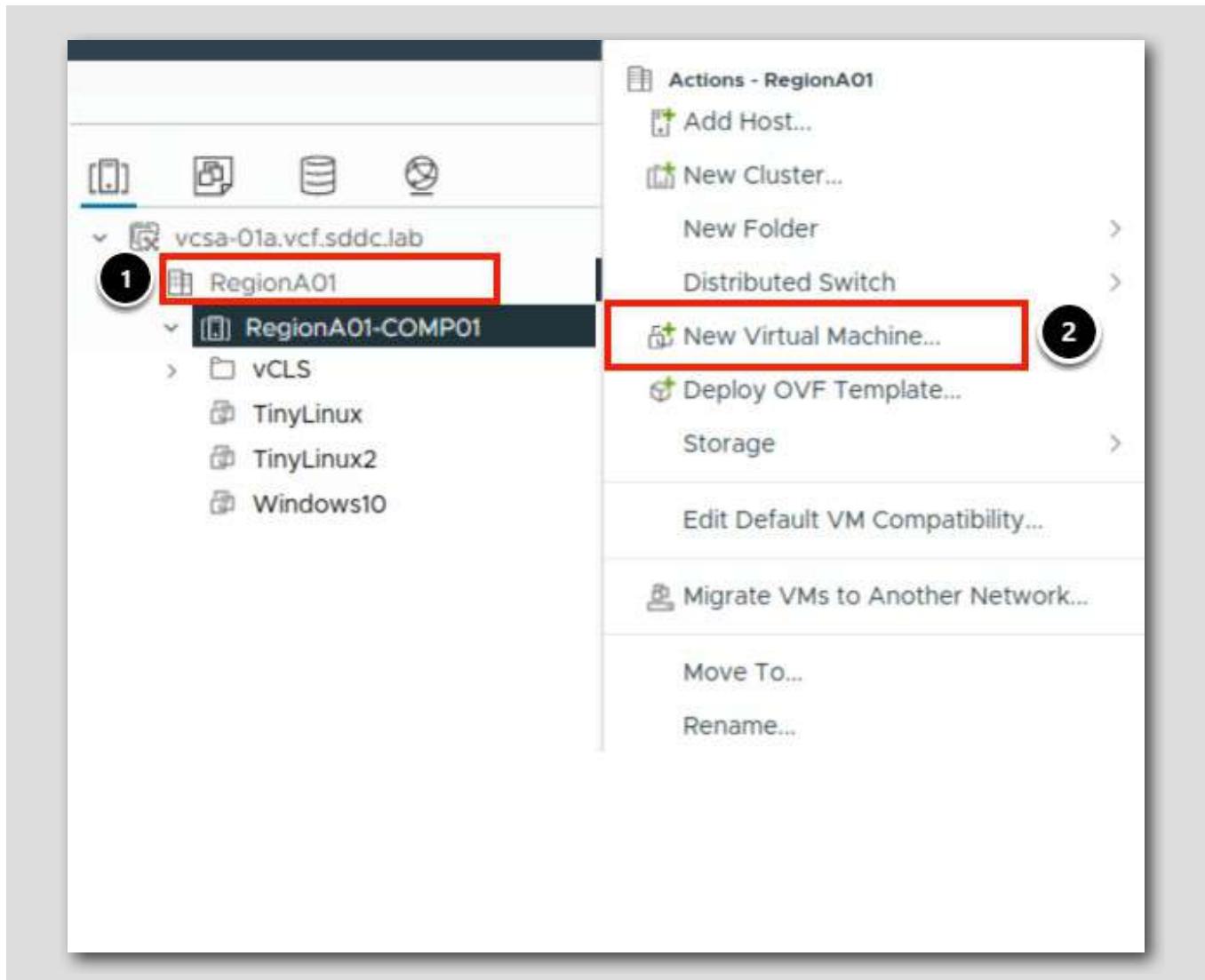
## Select and Expand Datacenter

[56]



1. Expand RegionA01 Datacenter if you cannot see the virtual machines under it can be seen.

## Start the New Virtual Machine Wizard



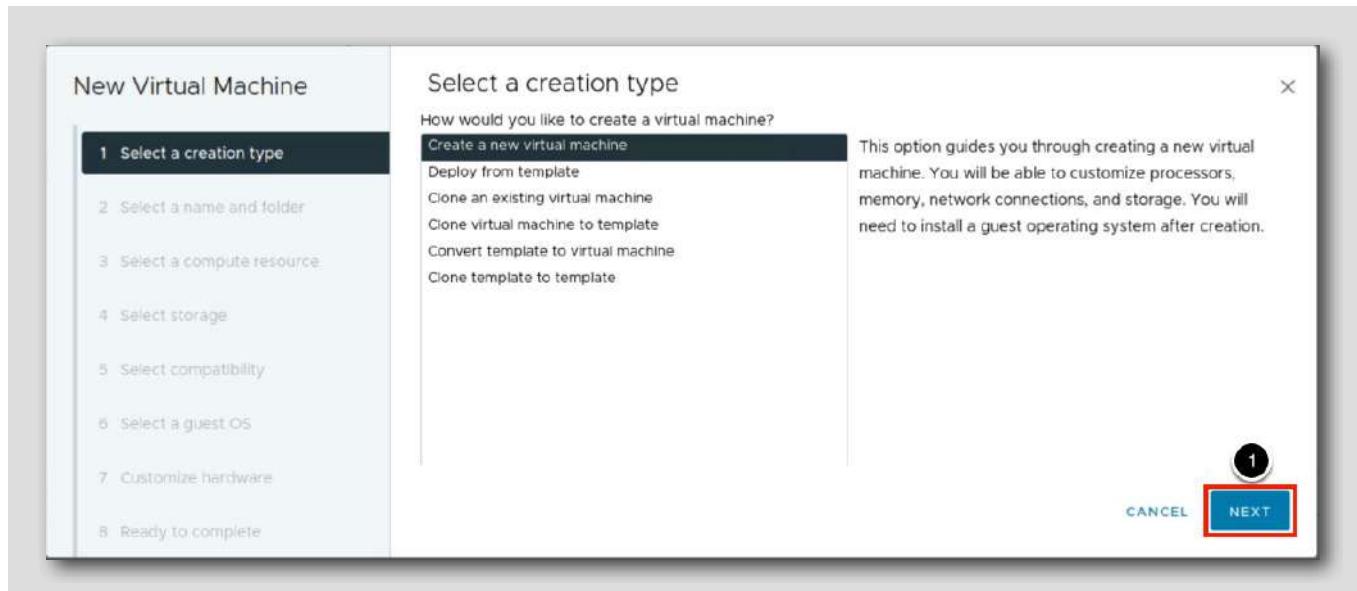
1. Right-click on RegionA01 Datacenter.

2. Click New Virtual Machine to start the new virtual machine wizard.

This wizard is used to create a new Virtual Machine and place it in the vSphere inventory.

## Virtual Machine Wizard

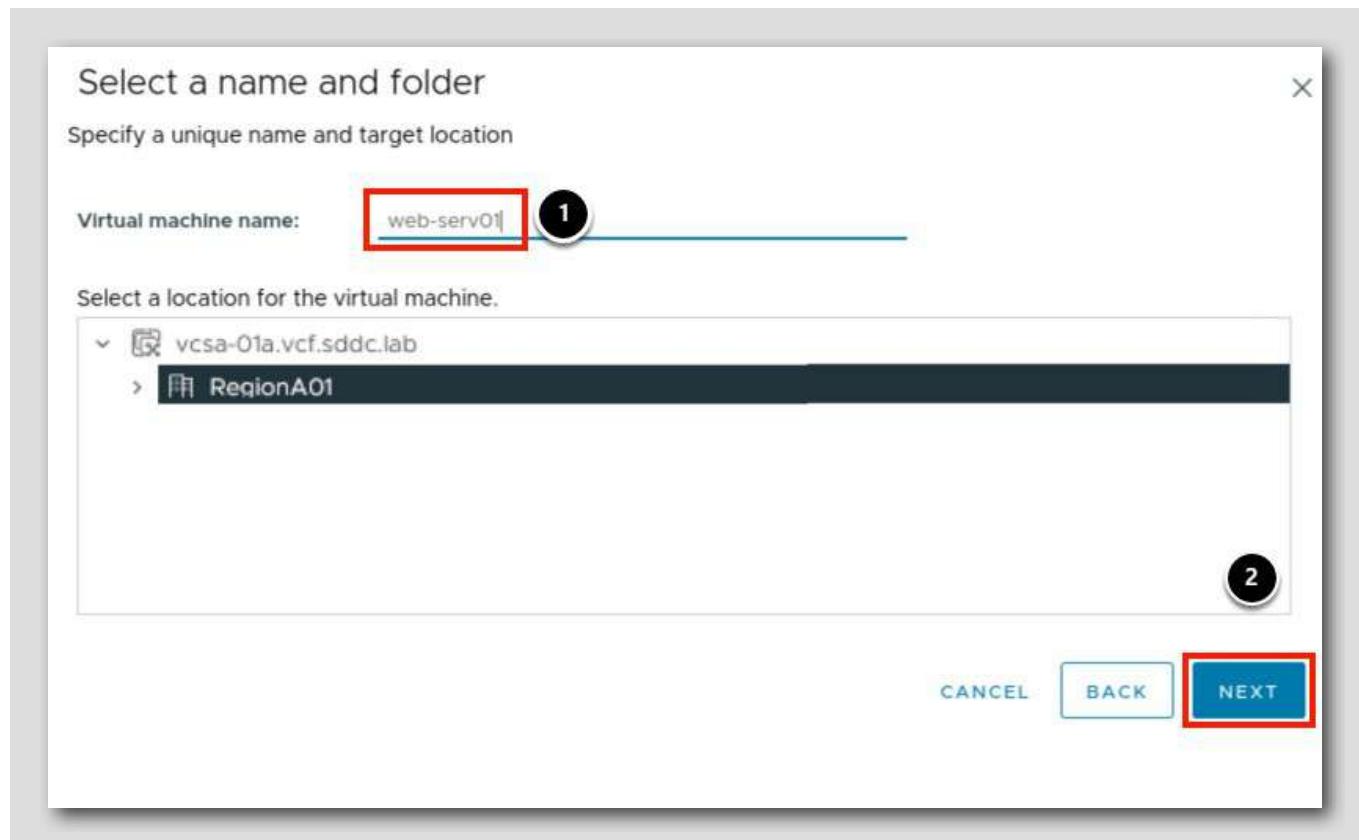
[58]



1. Since the Create a new virtual machine wizard is highlighted, just click Next.

## Name the Virtual Machine

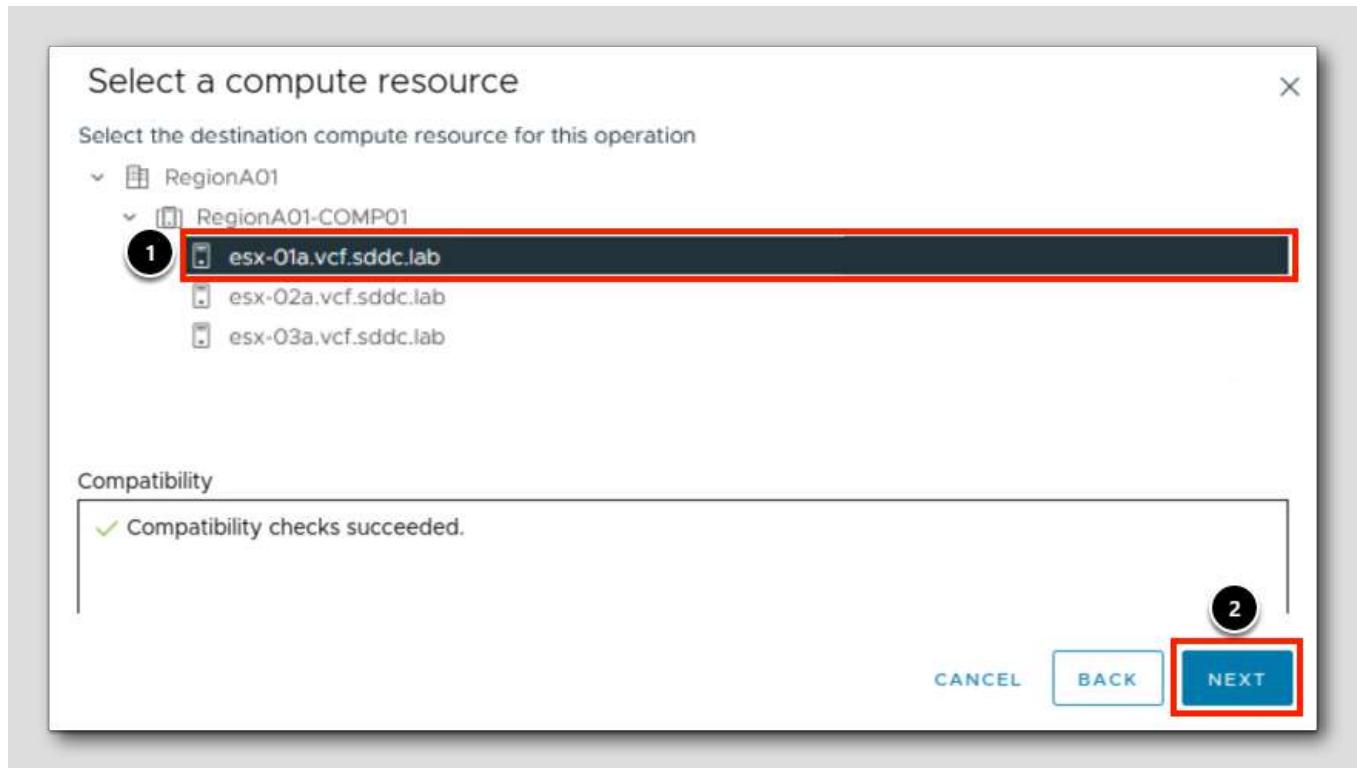
[59]



1. Enter web-serv01 for the name of the new virtual machine.

2. Click Next.

## Virtual Machine Placement



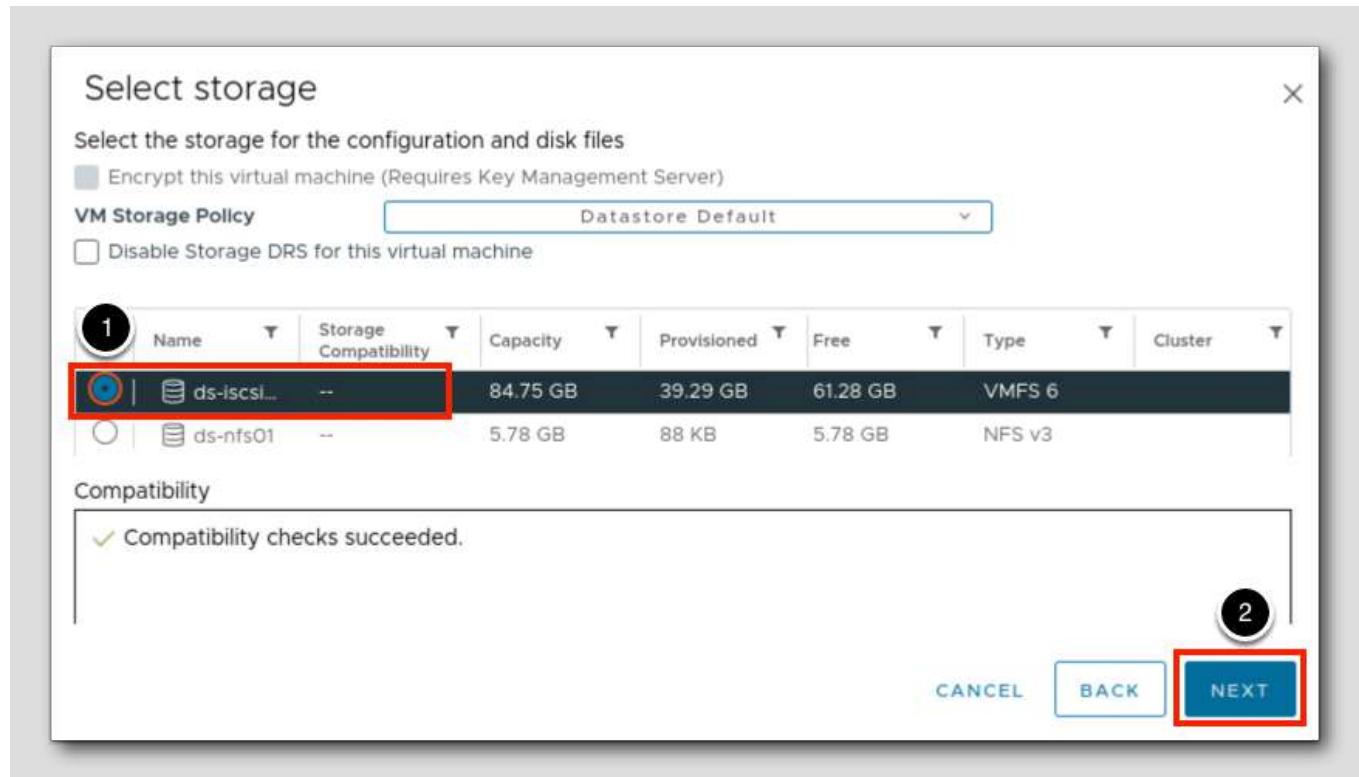
Because Distributed Resource Scheduler (DRS) is not enabled, you just have to select a host to use for the VM. More details on DRS will be covered later in this module.

1. Expand RegionA01 and RegionA01-COMP01, and select esx-01a.corp.vmbeans.com

2. Click Next.

## Select Storage

[61]

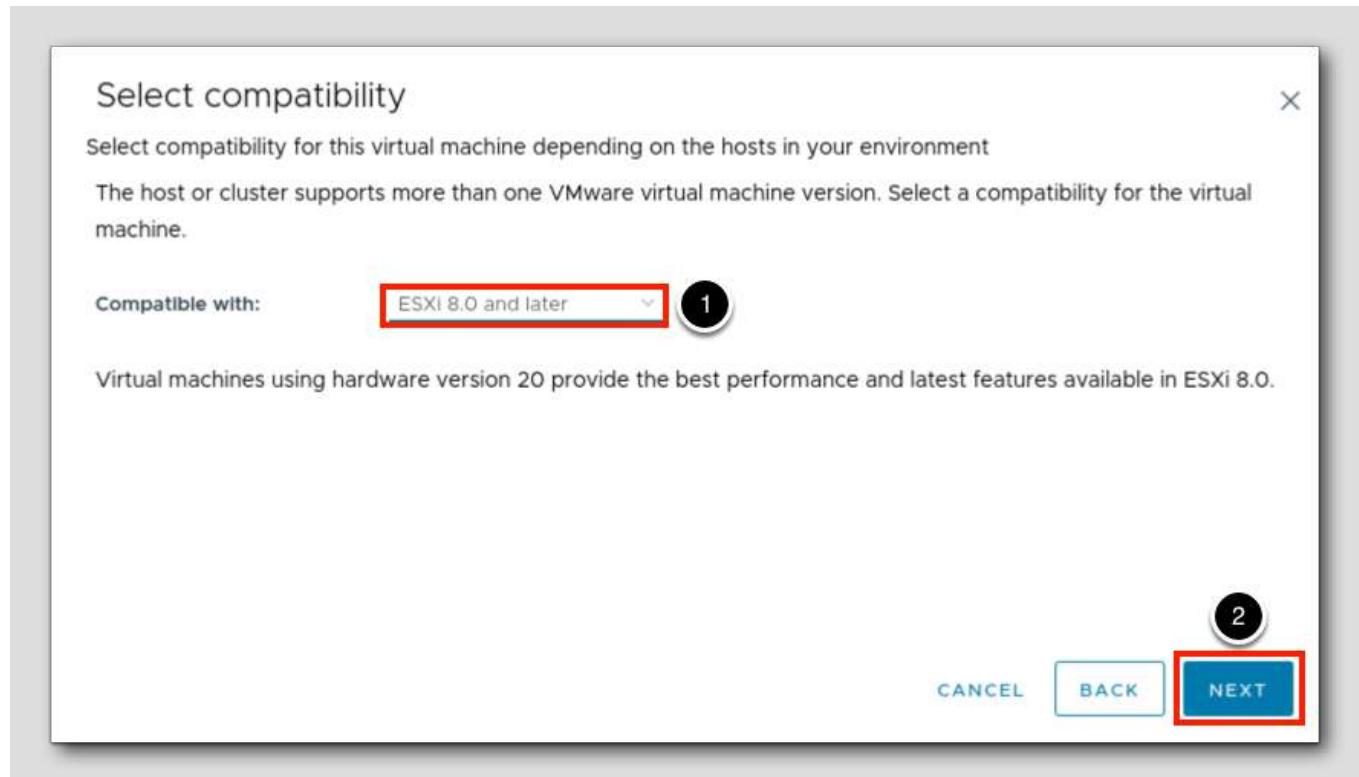


1. Select ds-iscsi01 datastore as the storage option.

2. Click Next.

## Compatibility

[62]

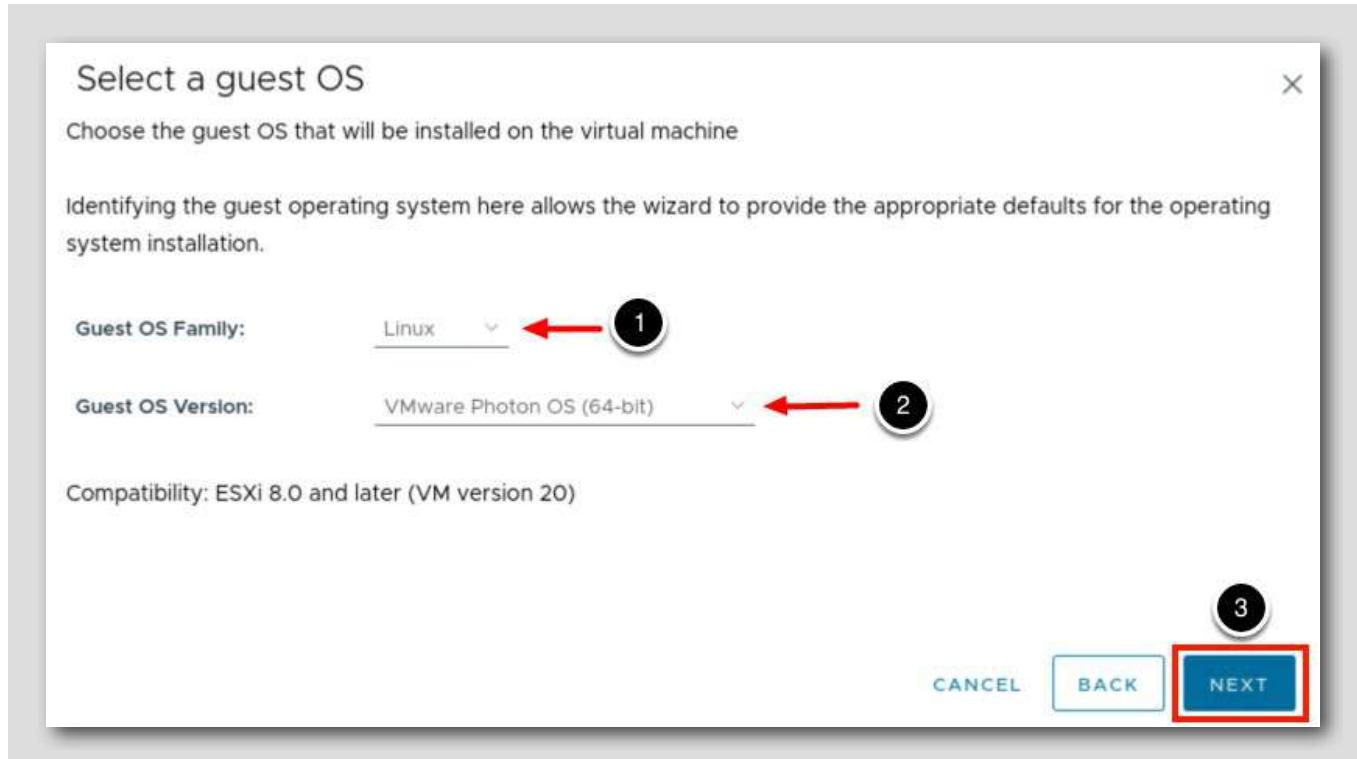


1. Select ESXi 8.0 and later.

2. Click Next to accept.

Guest OS

[63]

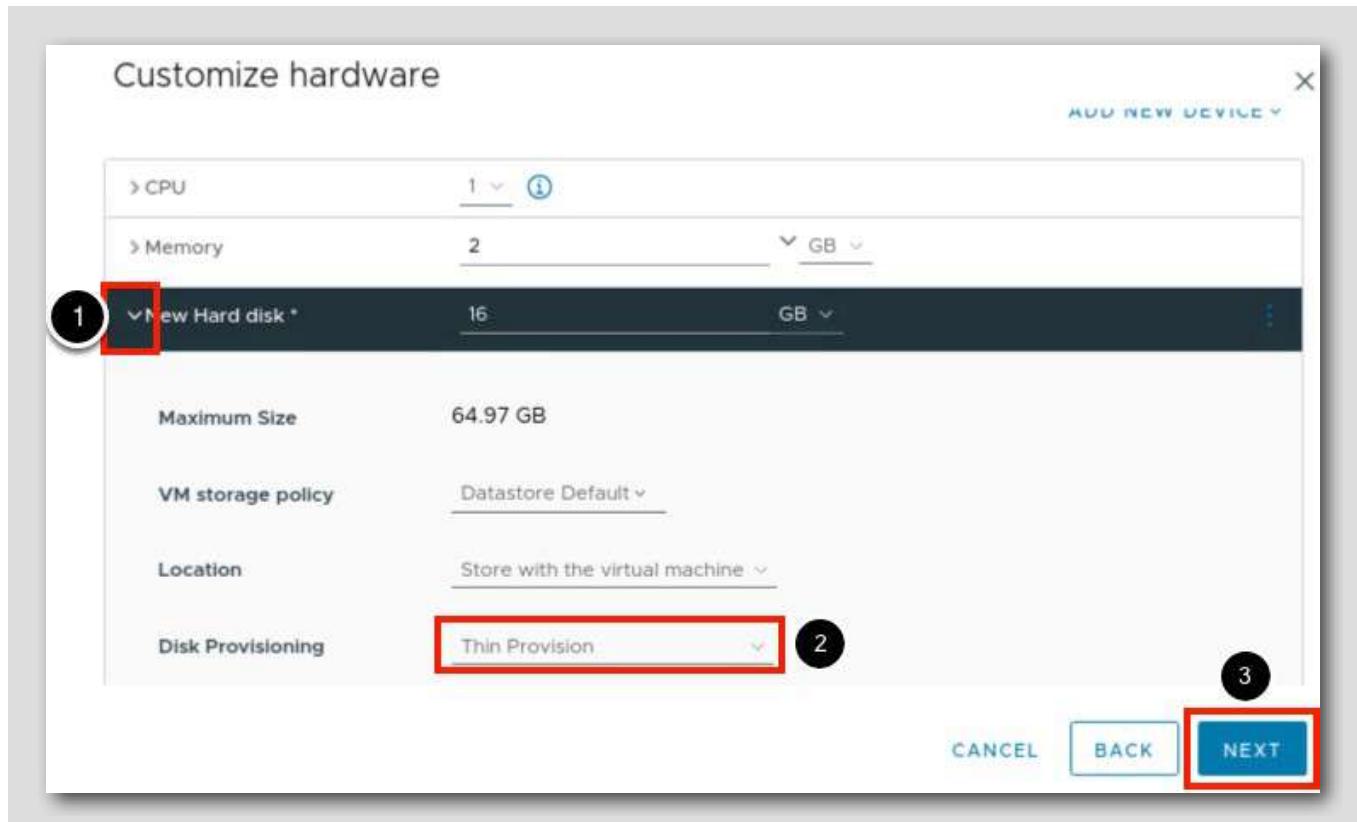


In this step, we will be selecting what operating system we will be installing. When we select the operating system, the supported virtual hardware and recommended configuration is used to create the virtual machine. Keep in mind this does not create a virtual machine with the operating system installed, but rather creates a virtual machine that is tuned appropriately for the operating system you have selected.

1. For the Guest OS Family, select Linux from the drop-down menu.
2. For the Guest OS Version, select VMware Photon OS (64-bit).
3. Click Next to continue.

## Customize Hardware

[64]



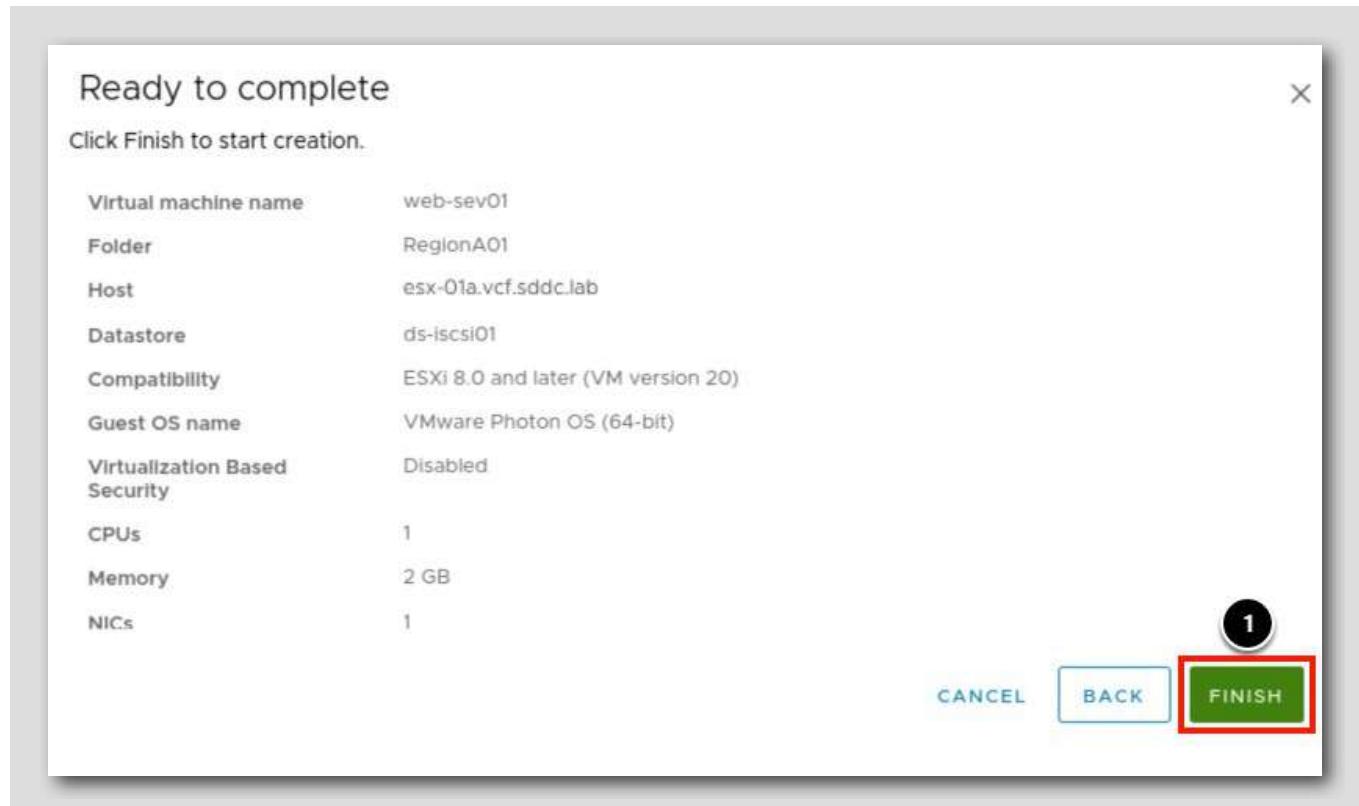
The recommended virtual hardware settings are shown as the default. These can be modified if needed.

1. Scroll down and expand the New Hard disk\* section.
2. For Disk Provisioning, select Thin Provision.
3. Click Next.

Note: if you miss choosing "Thin Provision", you will run out of disk space for this lab and your Virtual Machine may not power on. Disk space is limited in this lab environment.

## Ready to Complete

[65]

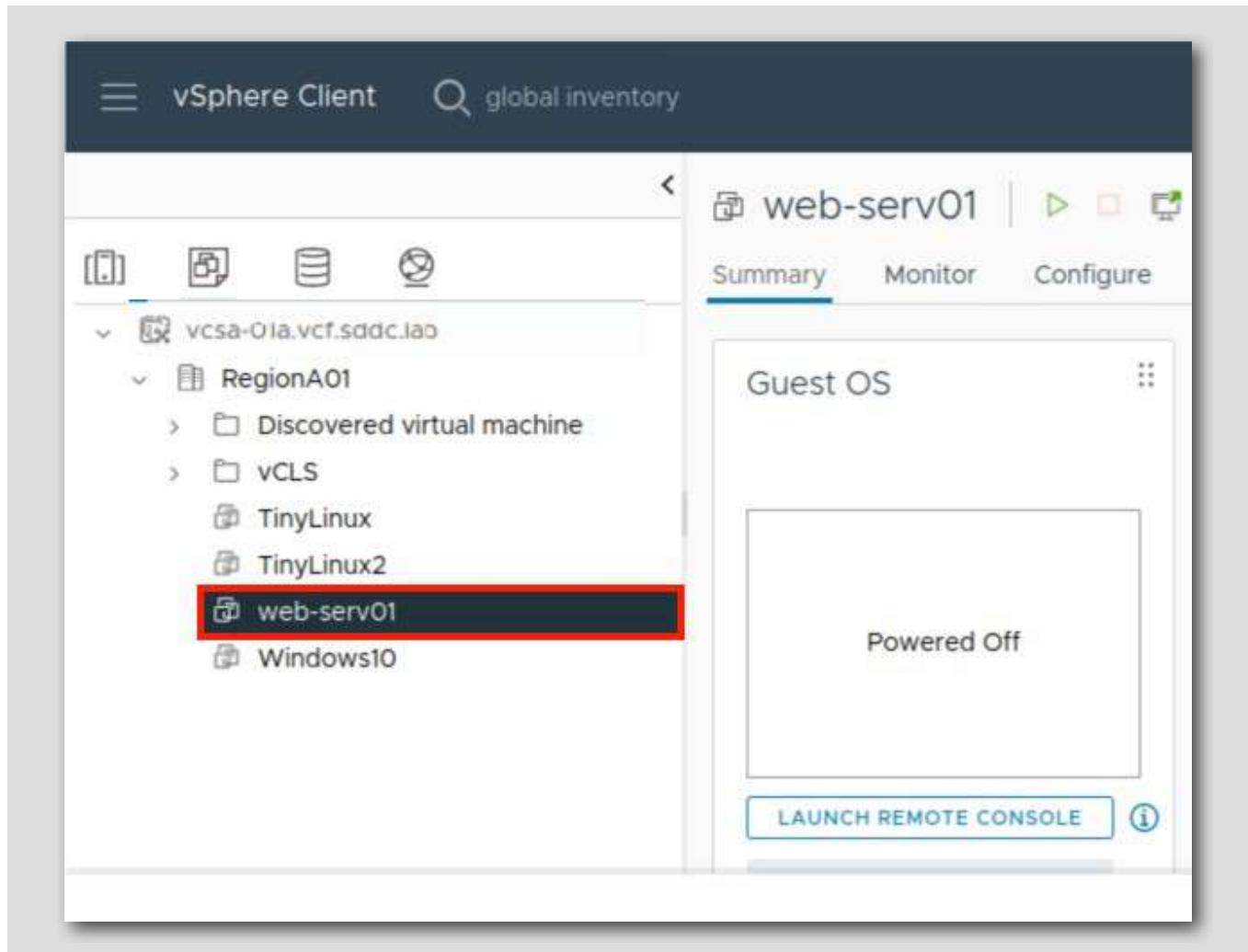


The settings for the virtual machine can be verified prior to it being created.

1. Click Finish to create the virtual machine.

## Newly Created Virtual Machine

[66]

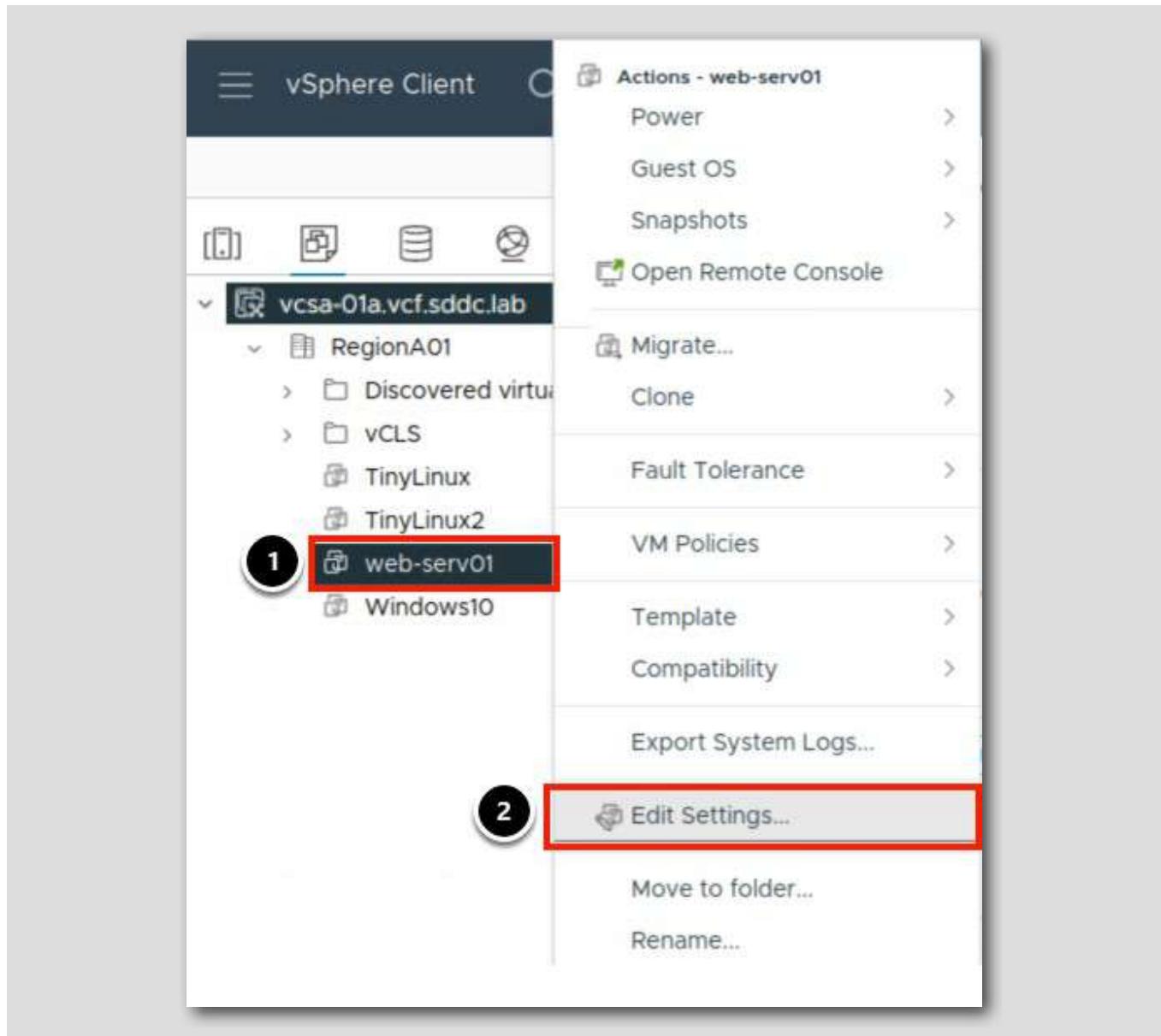


Congratulations on creating your first virtual machine web-serv01!

In the next steps, Photon OS will be installed on the virtual machine.

## Attaching an ISO to a Virtual Machine

[67]



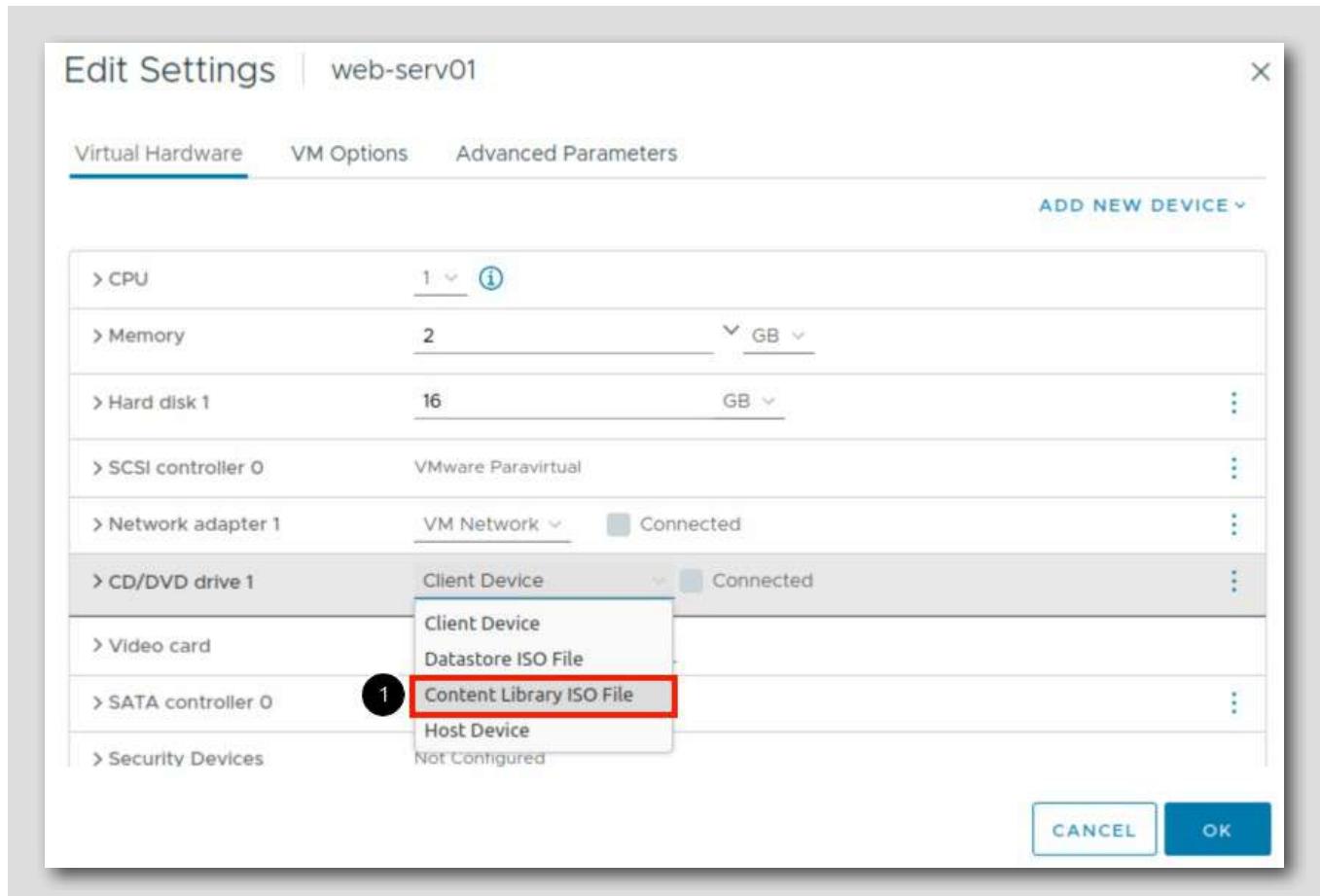
To make it easier to install operating systems on virtual machines, ISO images can be used. These can be kept in the same storage used for virtual machines. In addition, vCenter offers a Content Library as a repository. Content Libraries can then be synchronized to ensure every location is using the same versions.

1. To attach an ISO image to the virtual machine we just created, make sure web-serv01 is selected.

2.Right-click on web-serv01 and select Edit Settings...

## Content Library ISO File

[68]

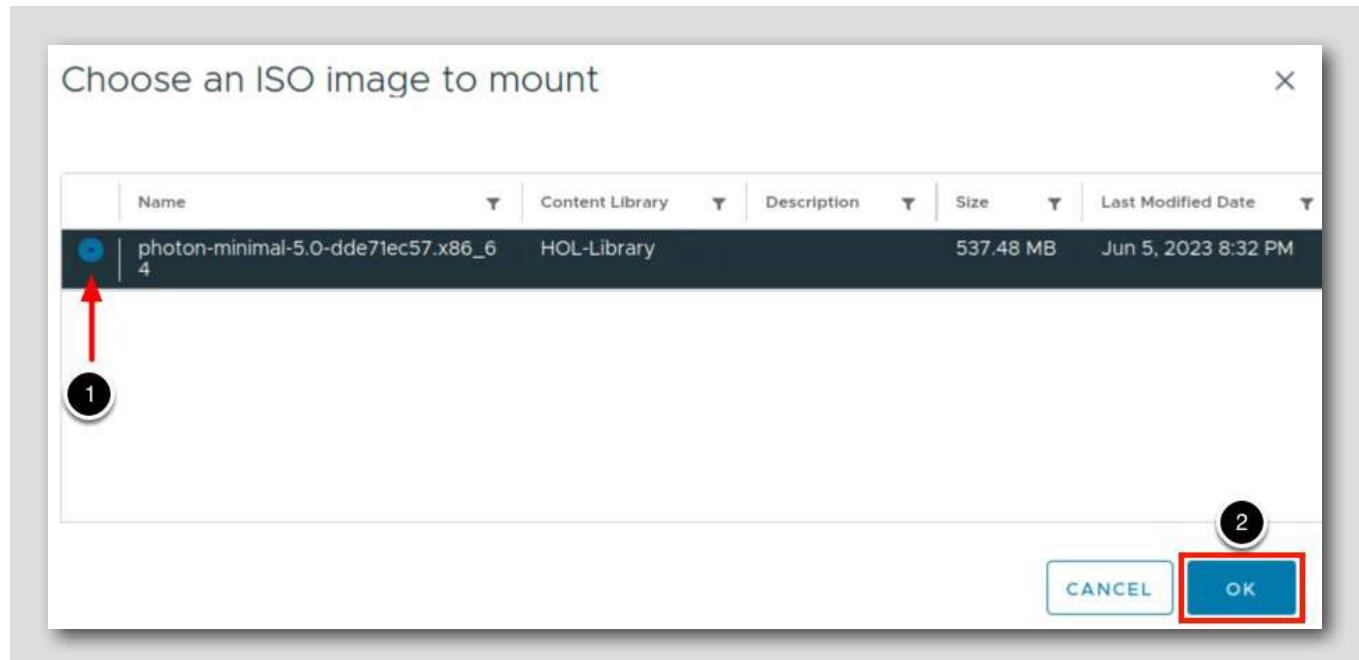


1. From the CD/DVD drive 1 drop-down menu, select Content Library ISO File.

This will open a file explorer to select that file.

## Select Photon

[69]

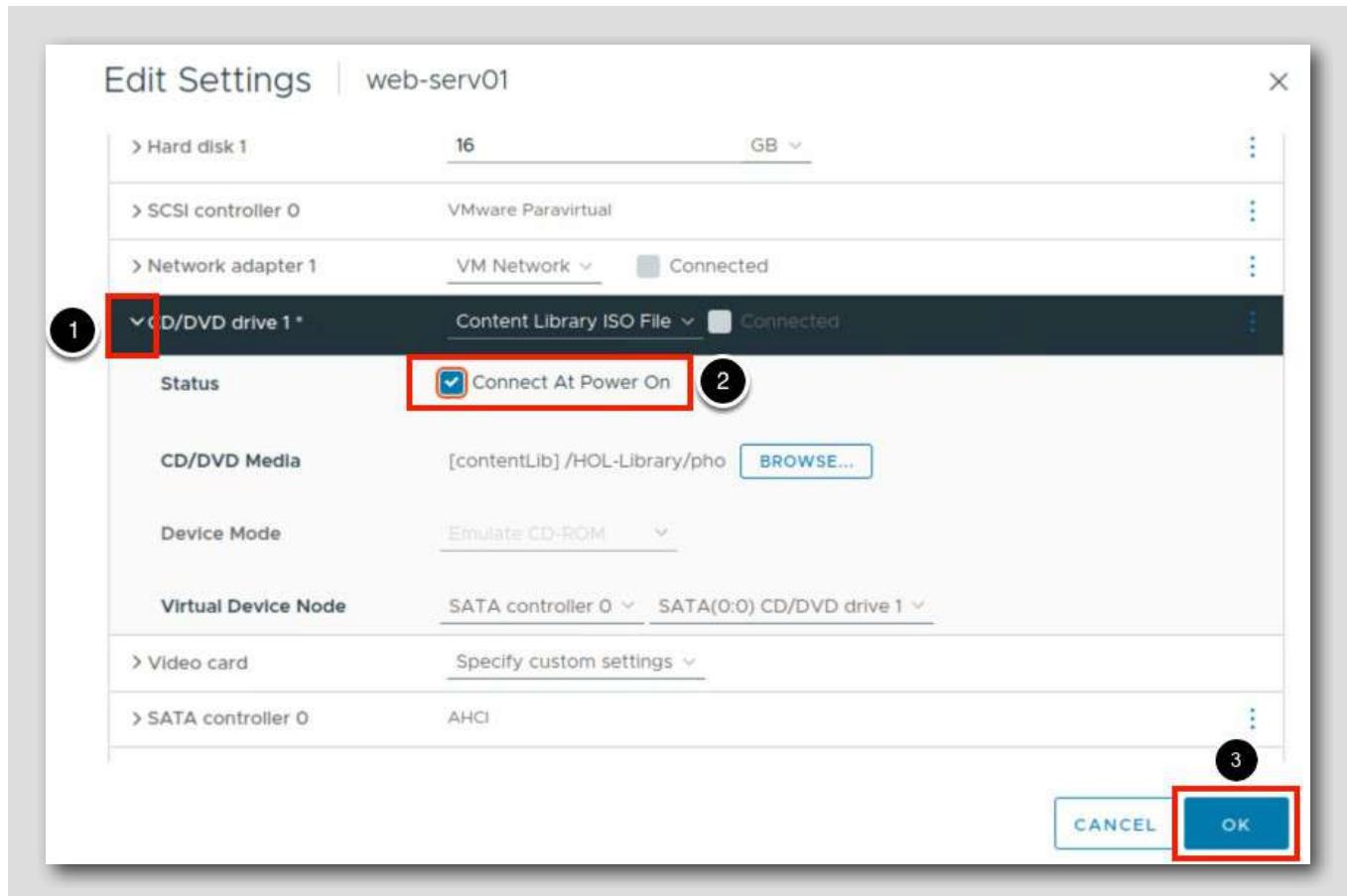


1. Click the radio button next to photon-minimal-5.0-dde71ec57.x86\_6

2. Click OK.

## Connect the Drive

[70]



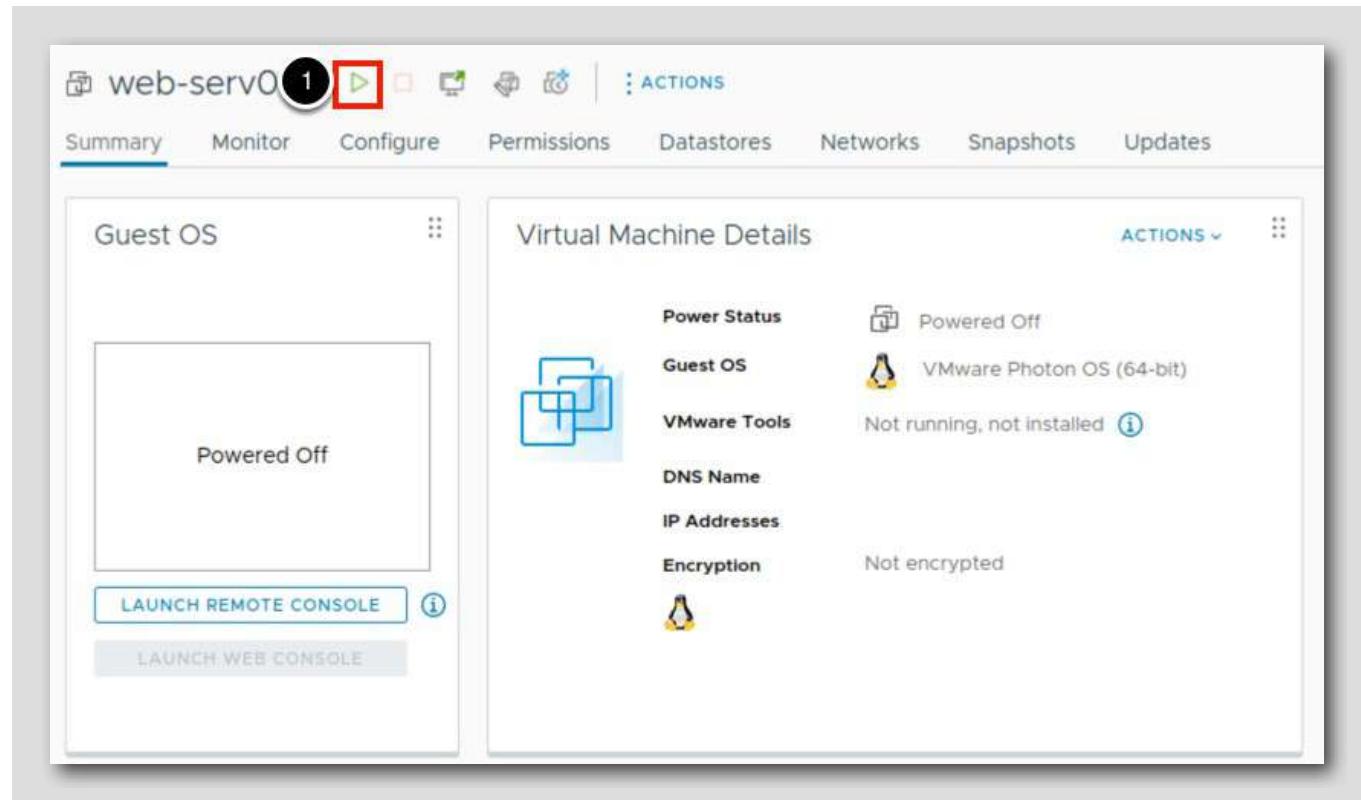
Finally, we want to attach or connect the ISO image to the virtual machine.

1. Expand the CD/DVD drive 1 menu
2. Click the Connect At Power On check box next to Status
3. Click OK.

Note: The virtual machine's BIOS is set to boot from CDROM only on first boot. If you miss this step you will need to manually adjust the BIOS to boot from CDROM just like a physical machine.

Power on web-serv01

[71]

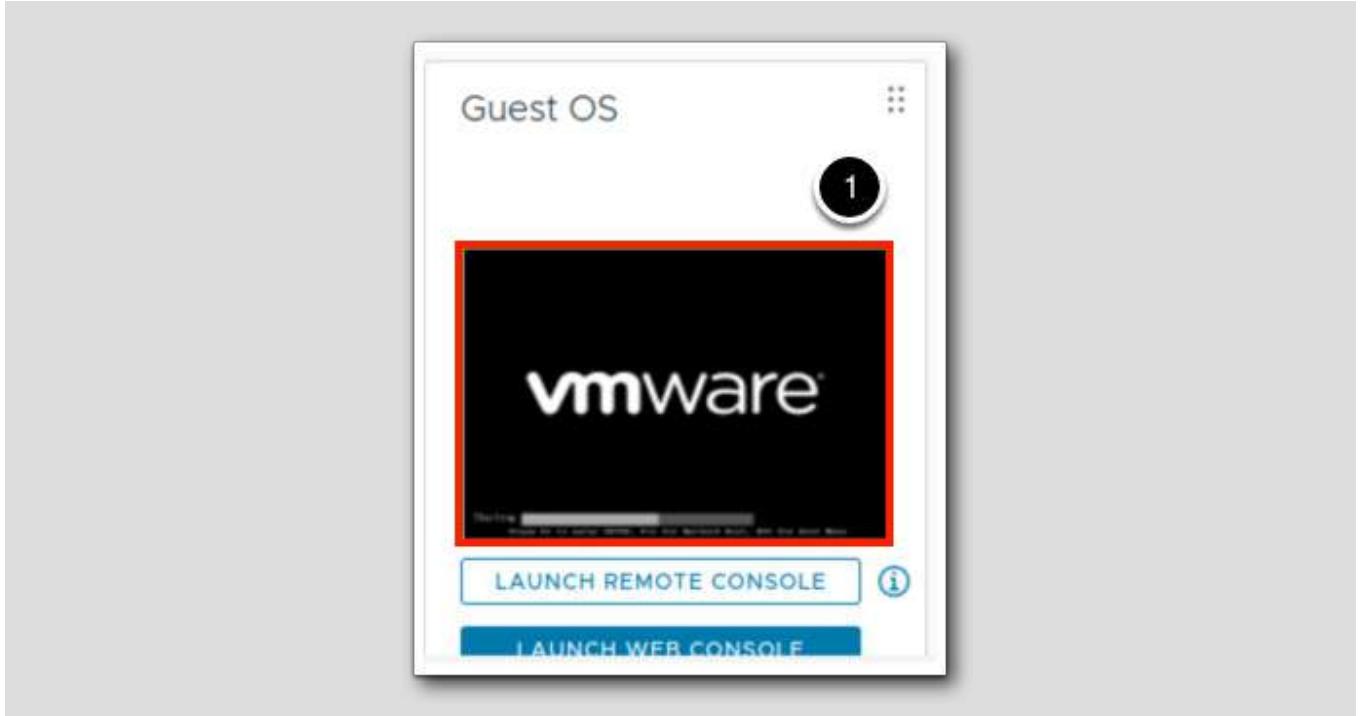


The screenshot shows the VMware vSphere Web Client interface. At the top, there's a navigation bar with tabs: Summary (underlined), Monitor, Configure, Permissions, Datastores, Networks, Snapshots, and Updates. A red box highlights the green play button icon in the top navigation bar. Below the navigation bar, there are two main sections: 'Guest OS' and 'Virtual Machine Details'. The 'Guest OS' section shows 'Powered Off'. The 'Virtual Machine Details' section includes fields for Power Status (Powered Off), Guest OS (VMware Photon OS (64-bit)), VMware Tools (Not running, not installed), DNS Name, IP Addresses, and Encryption (Not encrypted). There are also 'LAUNCH REMOTE CONSOLE' and 'LAUNCH WEB CONSOLE' buttons at the bottom of the left panel.

1. Click the green play button to power on the virtual machine and start the installation.

## Launch Console

[72]



1. To launch the console window, click anywhere in the console window screen. Alternatively, you can click on Launch Remote Console

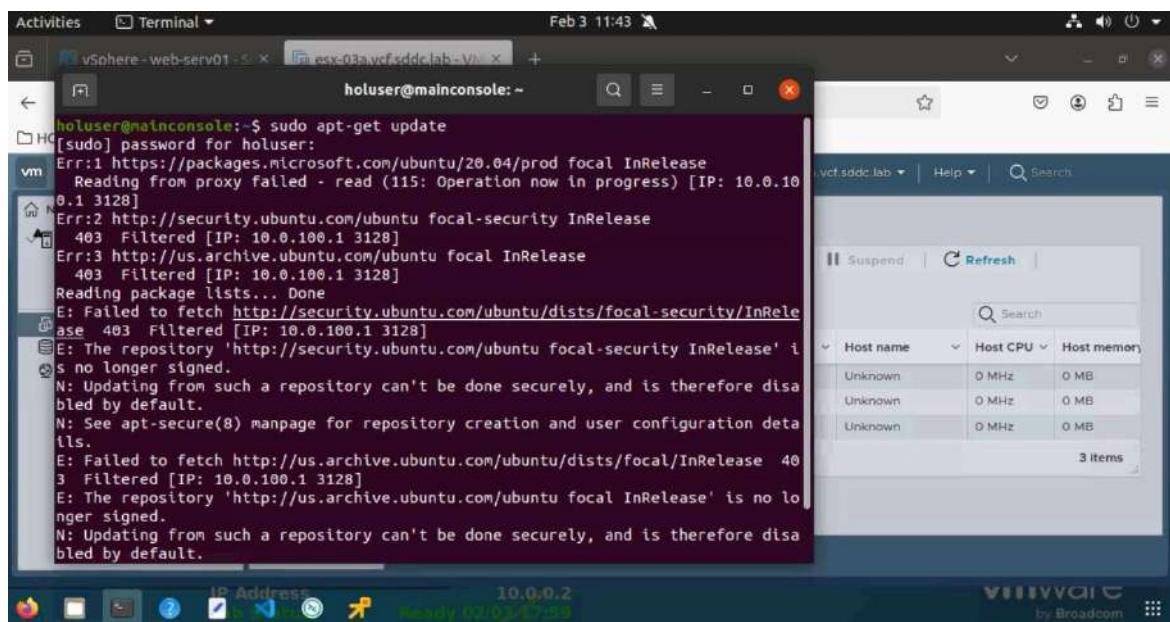
Note you also have the option of using the VMware Remote Console (VMRC). This is console is a separate application that needs to be installed on your local device as opposed to the Web Console which will launch in a new browser tab. The VMRC can be useful in certain situations when you need more capabilities, like attaching devices or power cycling options.

**Virtual Machine is created**

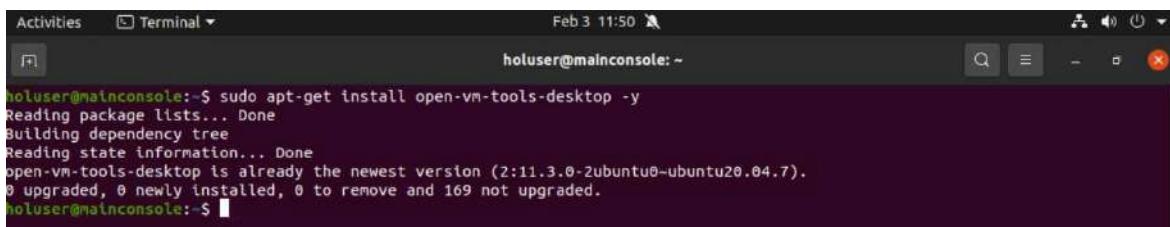
### Module 1.3 – Install VMware Tools

In VM, Open “Terminal” and follow the below steps:

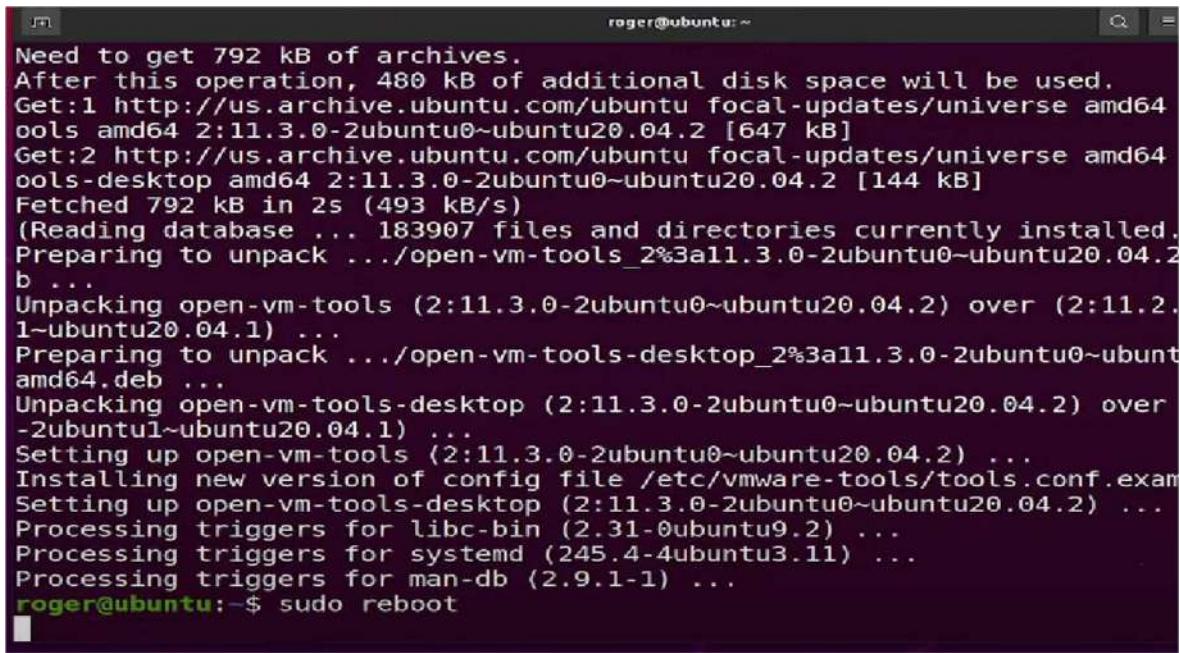
1. Run “Sudo apt-get update” command to updates the list of available packages and their versions stored in the system's package index as shown below:



2. Run “Sudo apt-get install open-vm-tools-desktop -y” command to install VMware Tools.



3. Run “Sudo reboot” to restart the system:



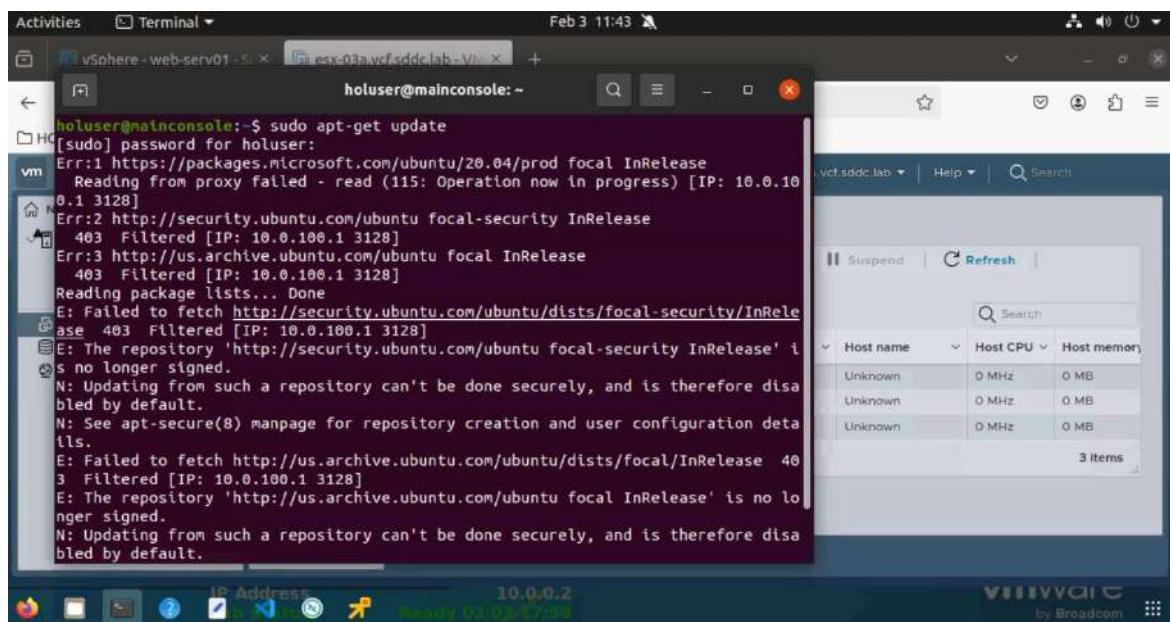
```
roger@ubuntu:~ Need to get 792 kB of archives.  
After this operation, 480 kB of additional disk space will be used.  
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64  
ools amd64 2:11.3.0-2ubuntu0~ubuntu20.04.2 [647 kB]  
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64  
ools-desktop amd64 2:11.3.0-2ubuntu0~ubuntu20.04.2 [144 kB]  
Fetched 792 kB in 2s (493 kB/s)  
(Reading database ... 183907 files and directories currently installed.  
Preparing to unpack .../open-vm-tools_2%3a11.3.0-2ubuntu0~ubuntu20.04.2  
b ...  
Unpacking open-vm-tools (2:11.3.0-2ubuntu0~ubuntu20.04.2) over (2:11.2.  
1-ubuntu20.04.1) ...  
Preparing to unpack .../open-vm-tools-desktop_2%3a11.3.0-2ubuntu0~ubunt  
amd64.deb ...  
Unpacking open-vm-tools-desktop (2:11.3.0-2ubuntu0~ubuntu20.04.2) over  
-2ubuntu1~ubuntu20.04.1) ...  
Setting up open-vm-tools (2:11.3.0-2ubuntu0~ubuntu20.04.2) ...  
Installing new version of config file /etc/vmware-tools/tools.conf.exam  
Setting up open-vm-tools-desktop (2:11.3.0-2ubuntu0~ubuntu20.04.2) ...  
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...  
Processing triggers for systemd (245.4-4ubuntu3.11) ...  
Processing triggers for man-db (2.9.1-1) ...  
roger@ubuntu:~$ sudo reboot
```

4. Once the System is rebooted, VMware Tools are installed.

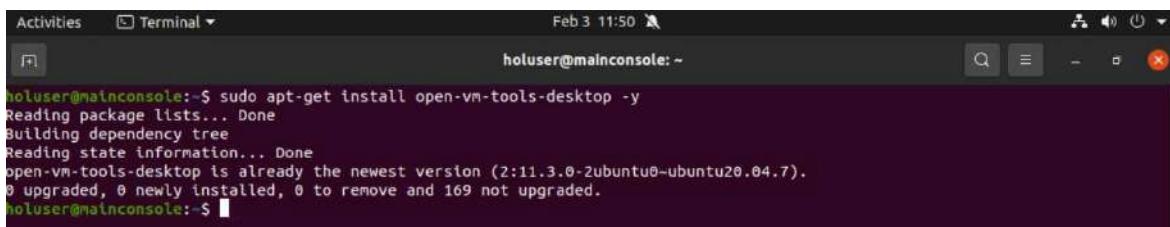
## Module 1.4 – Copy File to Desktop

In VM, Open “Terminal” and follow the below steps:

1. Run “Sudo apt-get update” command to updates the list of available packages and their versions stored in the system's package index as shown below:



2. Run “Sudo apt-get install open-vm-tools-desktop -y” command to install VMware Tools.



- Run “Sudo reboot” to restart the system:

```
roger@ubuntu:~$ Need to get 792 kB of archives.  
After this operation, 480 kB of additional disk space will be used.  
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64  
ools amd64 2:11.3.0-2ubuntu0~ubuntu20.04.2 [647 kB]  
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64  
ools-desktop amd64 2:11.3.0-2ubuntu0~ubuntu20.04.2 [144 kB]  
Fetched 792 kB in 2s (493 kB/s)  
(Reading database ... 183907 files and directories currently installed.  
Preparing to unpack .../open-vm-tools_2%3a11.3.0-2ubuntu0~ubuntu20.04.2  
b ...  
Unpacking open-vm-tools (2:11.3.0-2ubuntu0~ubuntu20.04.2) over (2:11.2.  
1~ubuntu20.04.1) ...  
Preparing to unpack .../open-vm-tools-desktop_2%3a11.3.0-2ubuntu0~ubunt  
amd64.deb ...  
Unpacking open-vm-tools-desktop (2:11.3.0-2ubuntu0~ubuntu20.04.2) over  
-2ubuntul~ubuntu20.04.1) ...  
Setting up open-vm-tools (2:11.3.0-2ubuntu0~ubuntu20.04.2) ...  
Installing new version of config file /etc/vmware-tools/tools.conf.exam  
Setting up open-vm-tools-desktop (2:11.3.0-2ubuntu0~ubuntu20.04.2) ...  
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...  
Processing triggers for systemd (245.4-4ubuntu3.11) ...  
Processing triggers for man-db (2.9.1-1) ...  
roger@ubuntu:~$ sudo reboot
```

- Once the System is rebooted, VMware Tools are installed.
- Copy Text/file from main Host machine and try pasting it in VM. You'll see paste option will be enabled.

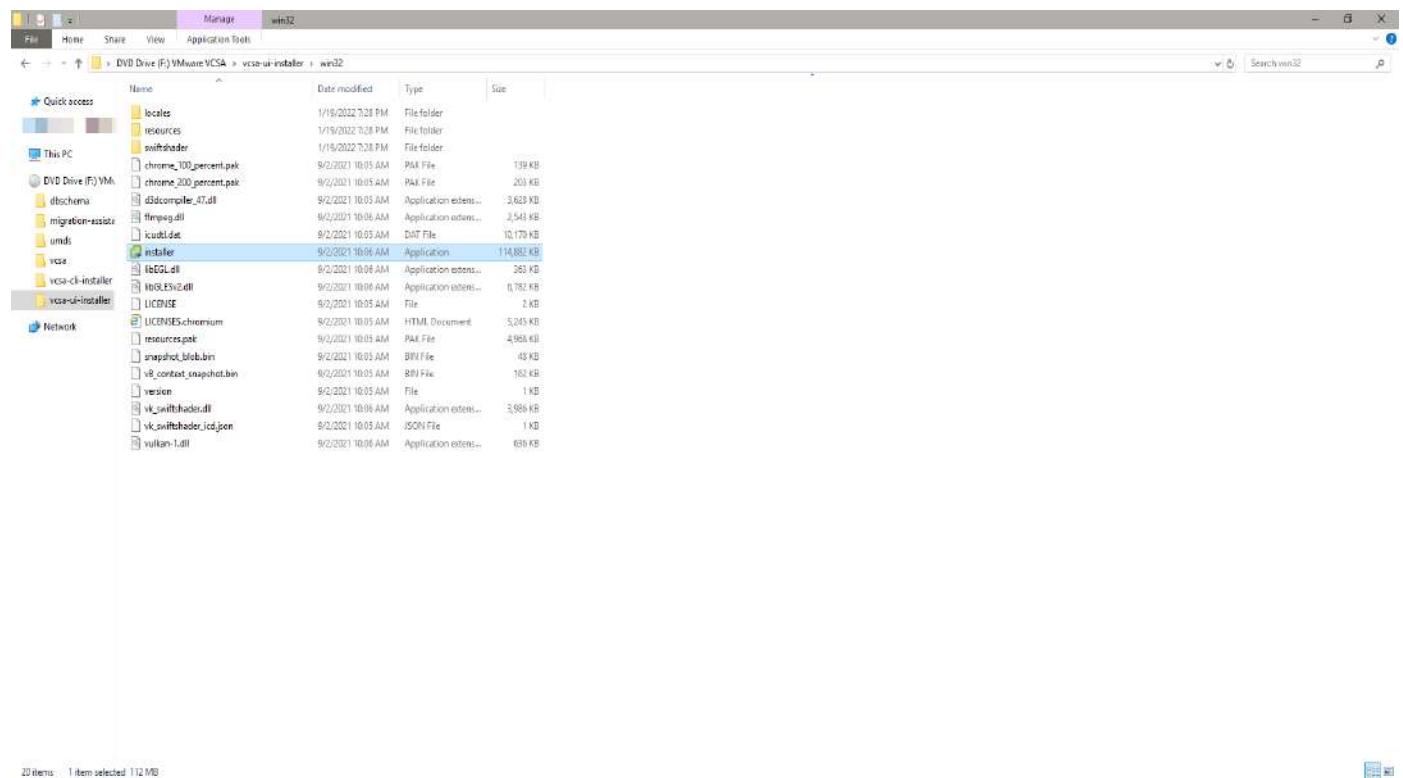


# Practical No 2 :- Working with vCenter Server Appliance

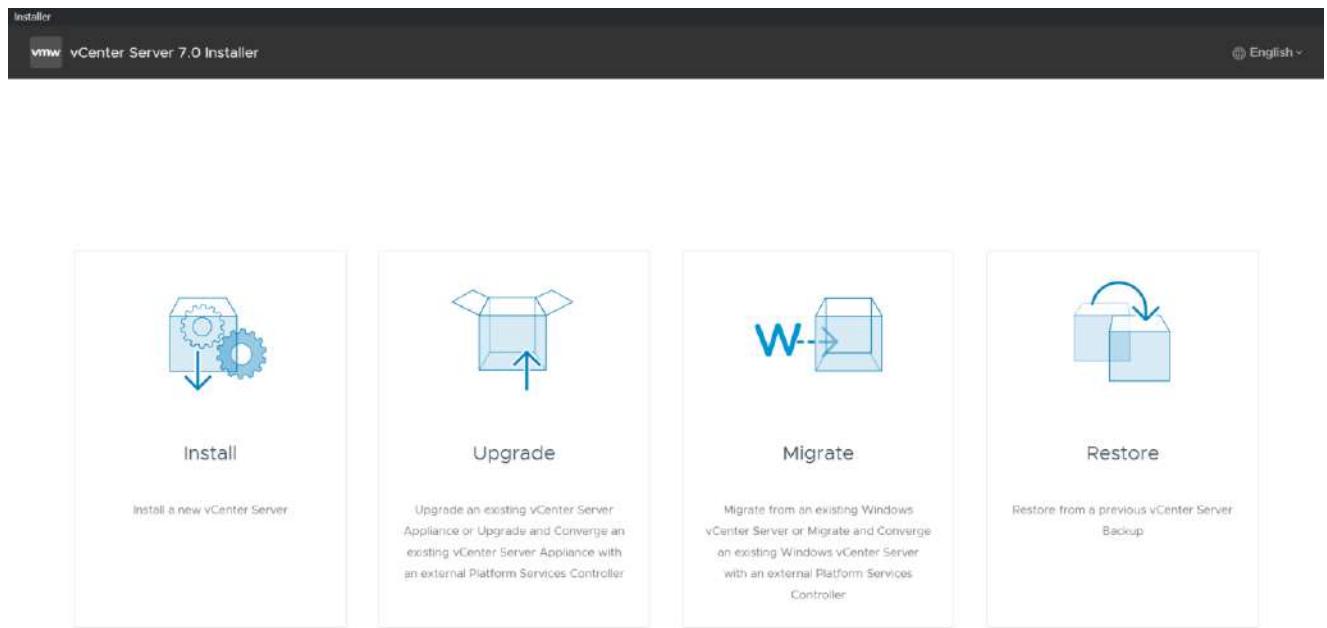
## 1. Access your vCenter Server Appliance and Configure Licenses

### vCenter Server Appliance (VCSA) Installation

1. Mount the ISO on your computer. The VCSA installer is compatible with Mac, Linux, and Windows.
2. Browse to the corresponding directory for your operating system, e.g., \vcsa-ui-installer\win32: right-click **Installer** and select **Run as administrator**.



3. As we are installing a new instance, click **Install**.



4. The install is two stages. We begin with Stage1: Deploy vCenter Server. Click **Next** to start.

The screenshot shows the "Install - Stage 1: Deploy vCenter Server" screen. The title bar says "vCenter Server 7.0 Installer" and "Install - Stage 1: Deploy vCenter Server".

**Introduction**

1 Introduction (selected)

2 End user license agreement

3 vCenter Server deployment target

4 Set up vCenter Server VM

5 Select deployment size

6 Select datastore

7 Configure network settings

8 Ready to complete stage 1

**Stage 1**

The External Platform Services Controller deployment has been deprecated. Learn more

This installer allows you to install a vCenter Server 7.0.

**Deploy vCenter Server**

Stage 2

**Set up vCenter Server**

Installing the vCenter Server is a two stage process. The first stage involves deploying a new vCenter Server to the target ESXi host or a compute resource in the target vCenter Server. The second stage completes the setup of the deployed vCenter Server. Click Next, to proceed with stage 1.

CANCEL NEXT

5. Read and accept the EULA, and then click **Next** to continue.

Installer

VMW Install - Stage 1: Deploy vCenter Server

1 Introduction

2 End user license agreement

3 vCenter Server deployment target

4 Set up vCenter Server VM

5 Select deployment size

6 Select datastore

7 Configure network settings

8 Ready to complete stage 1

End user license agreement

Read and accept the following license agreement.

VMWARE END USER LICENSE AGREEMENT

THE TERMS OF THIS END USER LICENSE AGREEMENT ("EULA") GOVERN YOUR USE OF THE SOFTWARE, REGARDLESS OF ANY TERMS THAT MAY APPEAR DURING THE INSTALLATION OF THE SOFTWARE.

BY DOWNLOADING, DEPLOYING, OR USING THE SOFTWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS EULA. IF YOU DO NOT AGREE TO THE TERMS OF THIS EULA, YOU MUST NOT DOWNLOAD, DEPLOY, OR USE THE SOFTWARE, AND YOU MUST DELETE OR RETURN THE UNUSED SOFTWARE TO US OR THE VMWARE CHANNEL PARTNER FROM WHICH YOU ACQUIRED IT WITHIN THIRTY (30) DAYS OF ITS ACQUISITION AND REQUEST A REFUND OF THE LICENSE FEE, IF ANY, THAT YOU PAID FOR THE SOFTWARE.

EVALUATION LICENSE. If you license the Software for evaluation purposes (an "Evaluation License"), your use of the Software is only permitted for a period of thirty (30) days (unless we specify otherwise), and you may not use the Software with production data. Notwithstanding any other provision in this EULA, an Evaluation License of the Software is provided "AS IS" without indemnification, support or warranty of any kind, express or implied.

LICENCE GRANT

I accept the terms of the license agreement.

CANCEL BACK NEXT

6. Select the ESXi host on which to install the VCSA as a guest. This must be a host that runs ESXi 6.5 or later. NVIDIA recommends that the vCenter server (Windows or appliance-based) run on a separate management cluster from the one designated for VDI workloads. Enter the IP address or fully qualified domain name (FQDN) of the chosen host, its root username, and password; then click **Next**.

Installer

VMW Install - Stage 1: Deploy vCenter Server

1 Introduction

2 End user license agreement

3 vCenter Server deployment target

4 Set up vCenter Server VM

5 Select deployment size

6 Select datastore

7 Configure network settings

8 Ready to complete stage 1

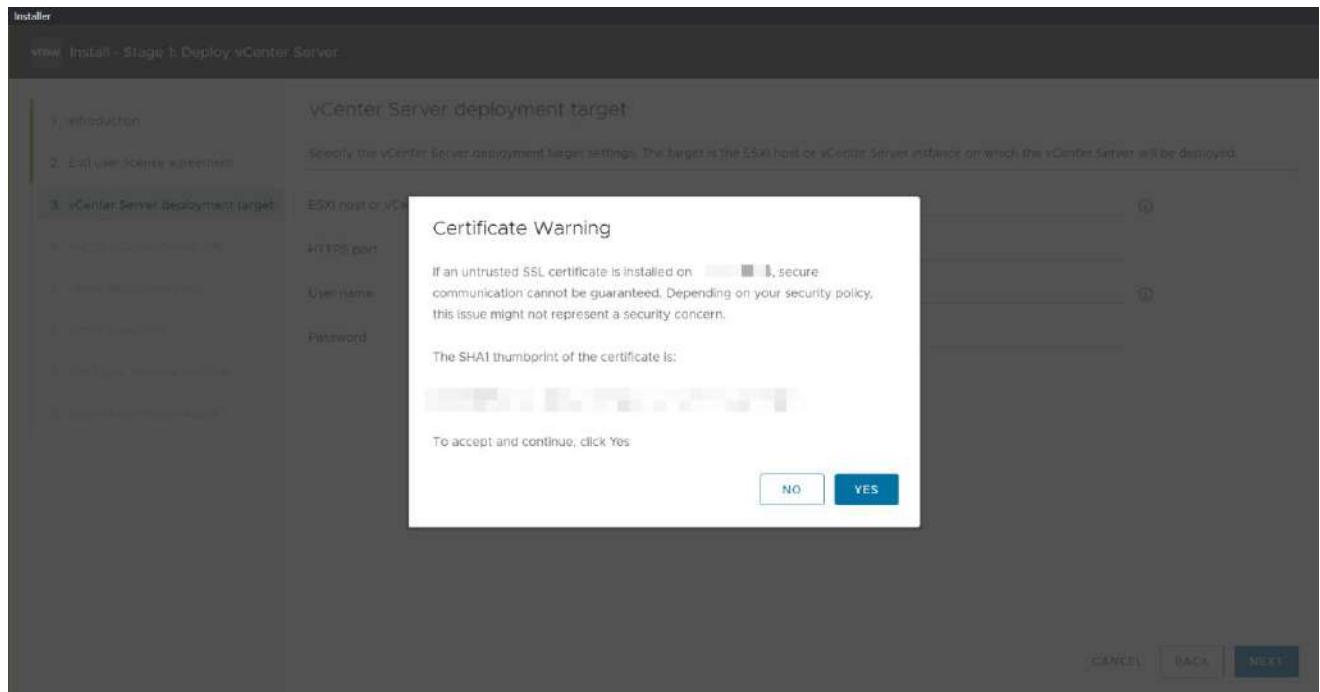
vCenter Server deployment target

Specify the vCenter Server deployment target settings. The target is the ESXi host or vCenter Server instance on which the vCenter Server will be deployed.

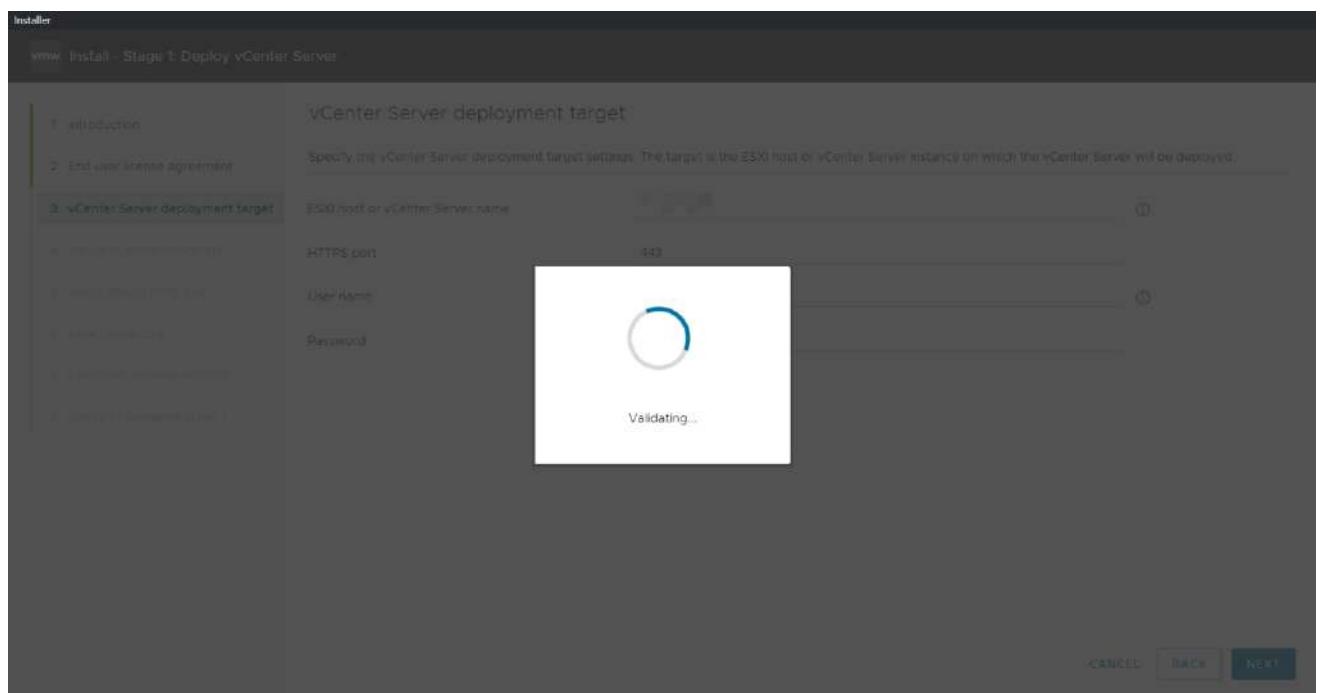
ESXi host or vCenter Server name	.....	①
HTTPS port	443	
User name	root	②
Password	*****	

CANCEL BACK NEXT

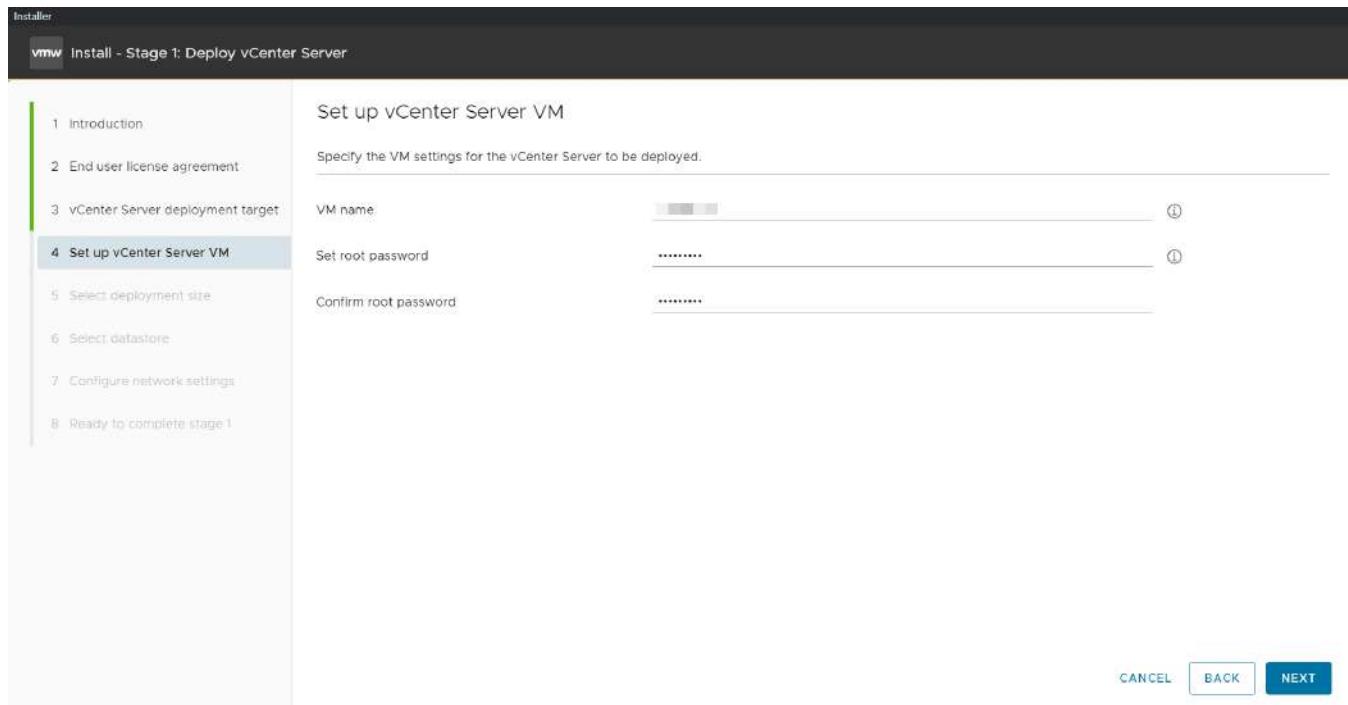
7. If your desktop can reach the host, you should see a certificate warning as it connects. This warning is due to the use of a self-signed certificate. If you are using a signed certificate, you will not see this warning. Click **Yes** to continue.



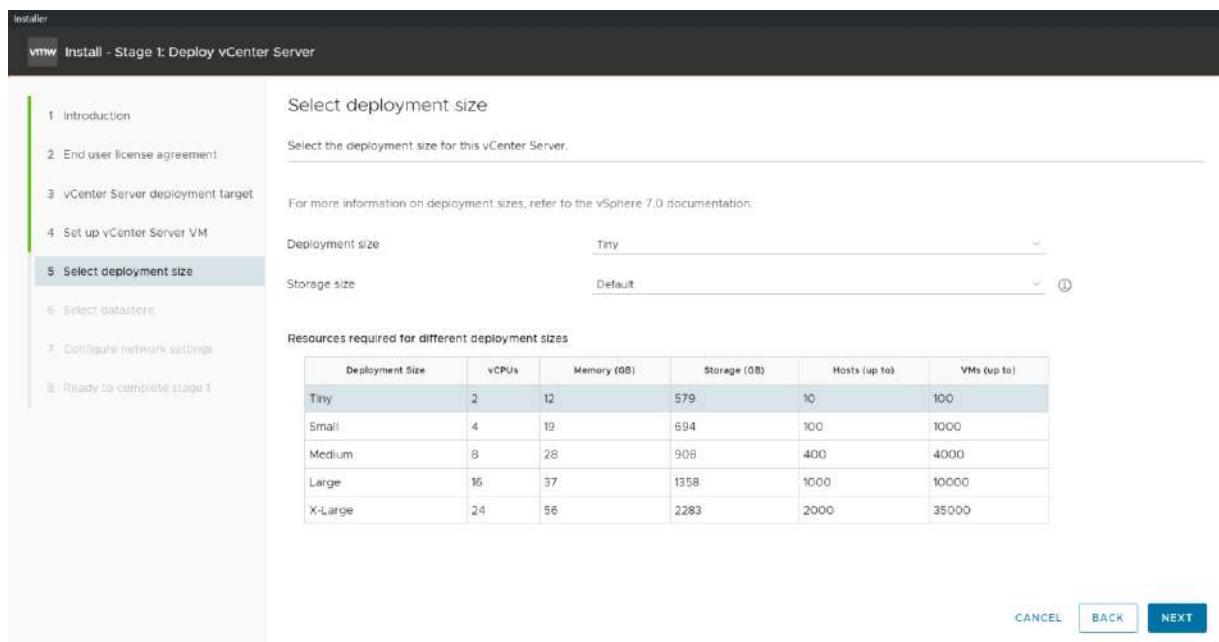
8. The credentials are validated.



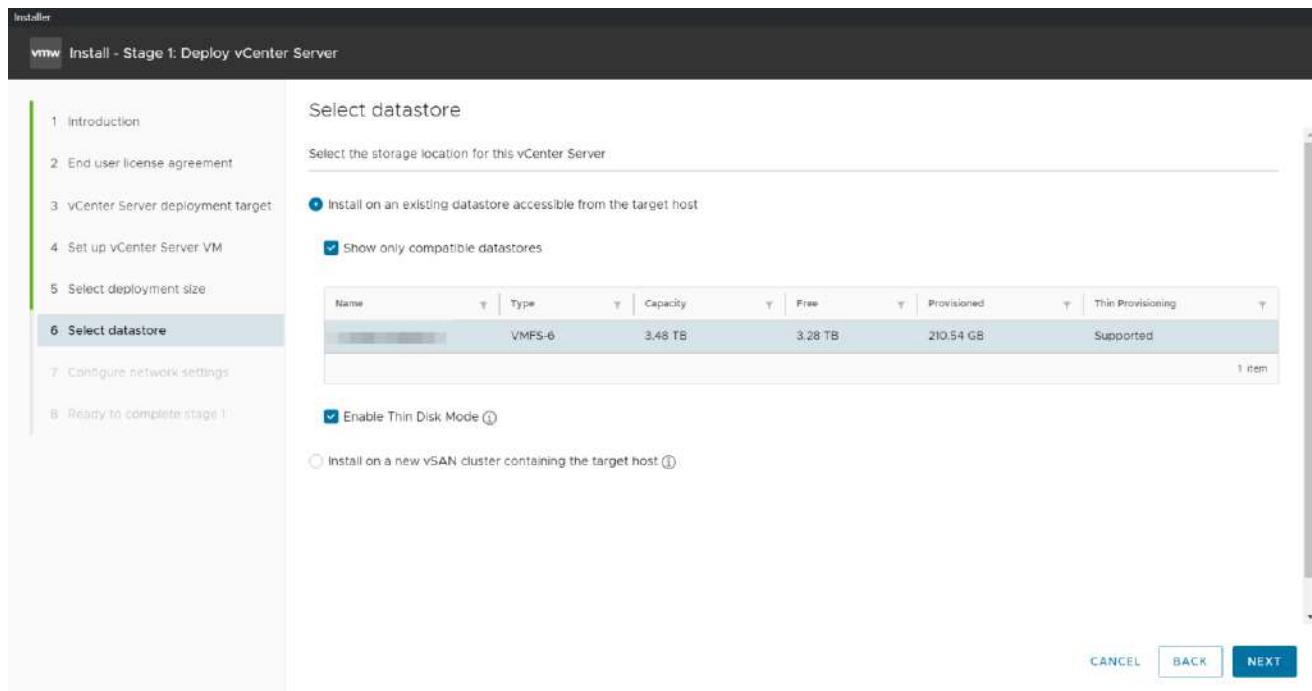
9. When prompted after a successful connection, provide a VM name for the vCenter Server 7, type the passwords in the Set Root password field, and enter the root password again, and click **Next**.



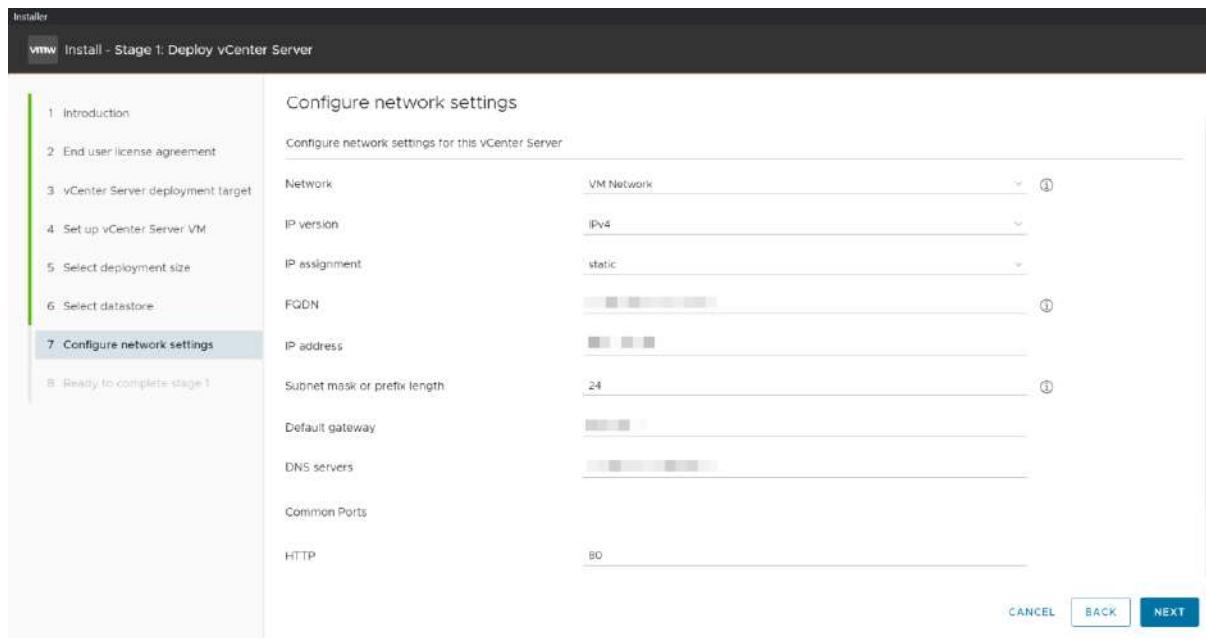
10. Select a deployment size appropriate to the number of hosts and virtual machines that vCenter Server will manage, then click **Next**.



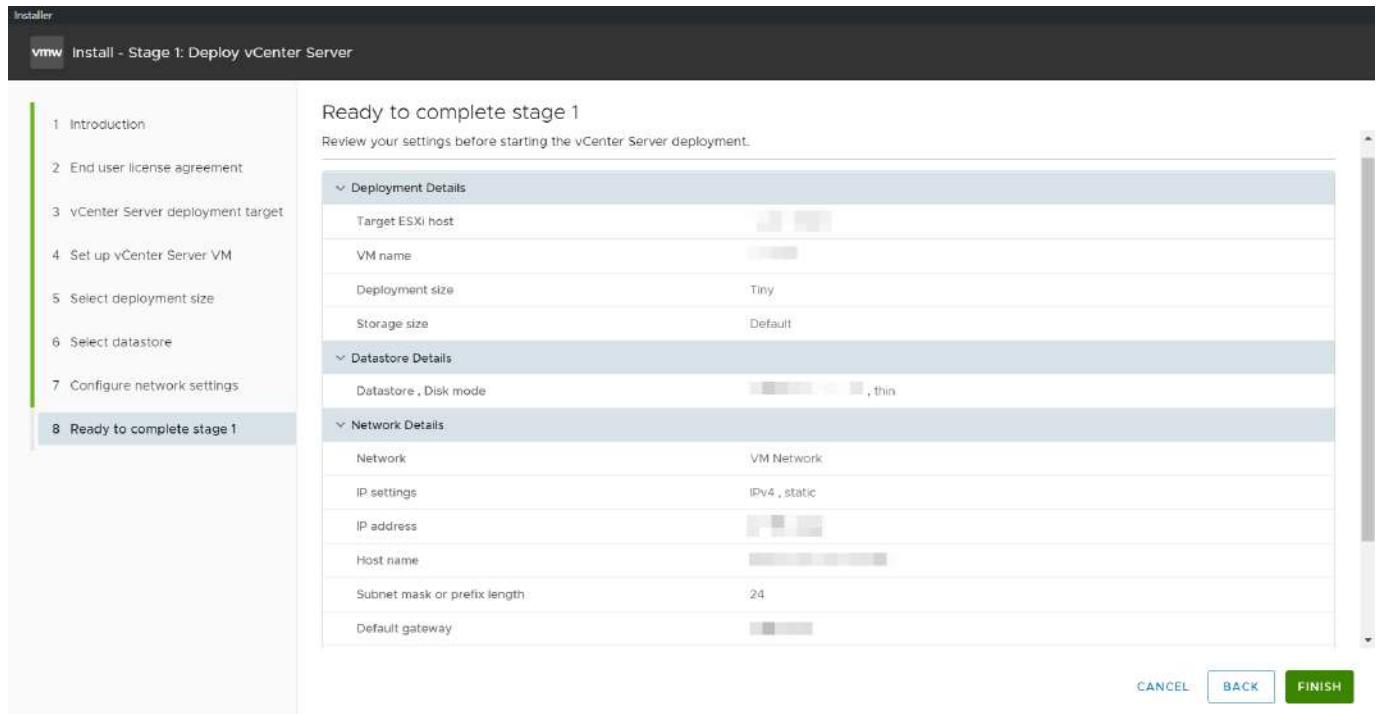
11. Select the datastore where the VCSA will be deployed, select thin provisioning if required, and click **Next**.



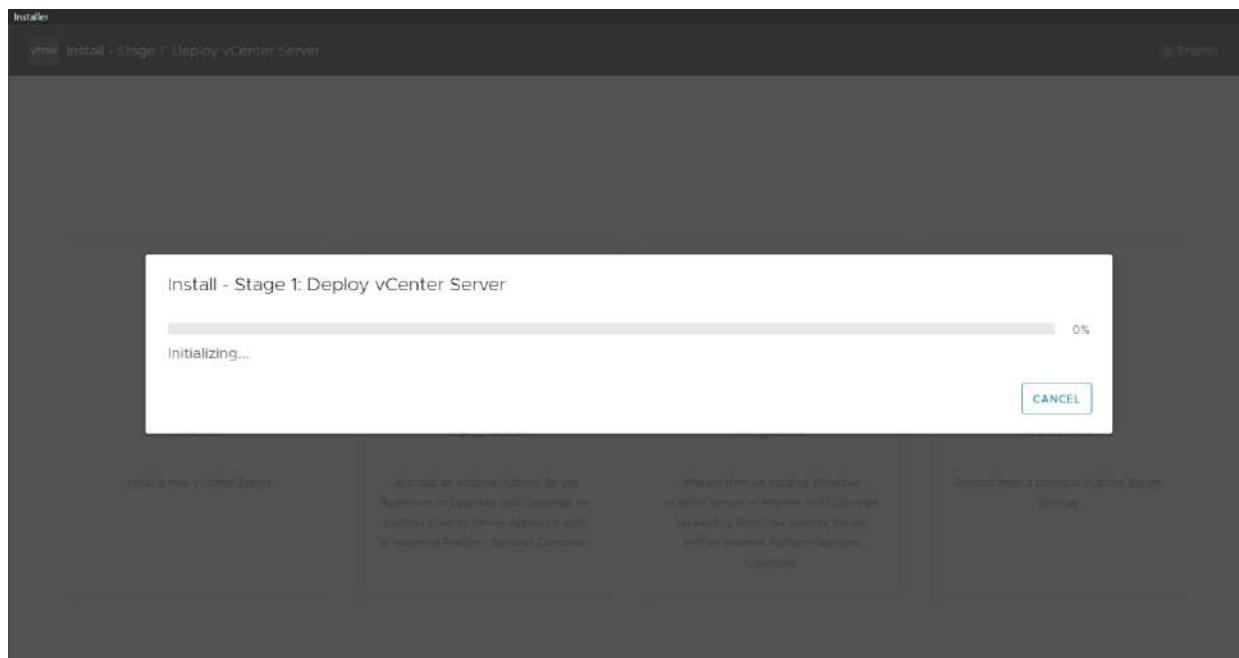
12. The installer displays the **Configure network settings**. Before you configure these settings, choose an appropriate static IP address, and enter it into local DNS (e.g., on the Domain Controller). Once you can resolve the address, enter the IP address hostname on the network setting page, then scroll down and enter the remaining items. When all desired settings are complete, select **Next**.



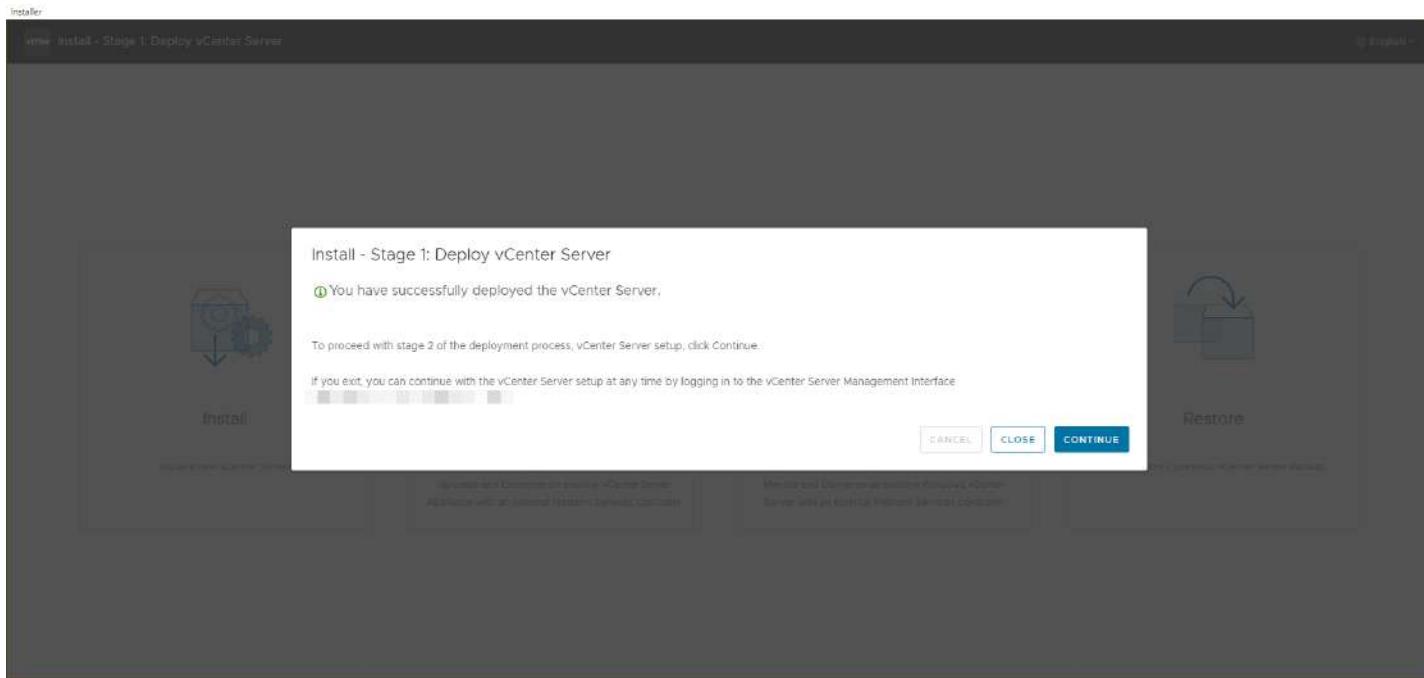
13. Review the settings before starting the vCenter Server deployment and click **Finish** to start the installation.



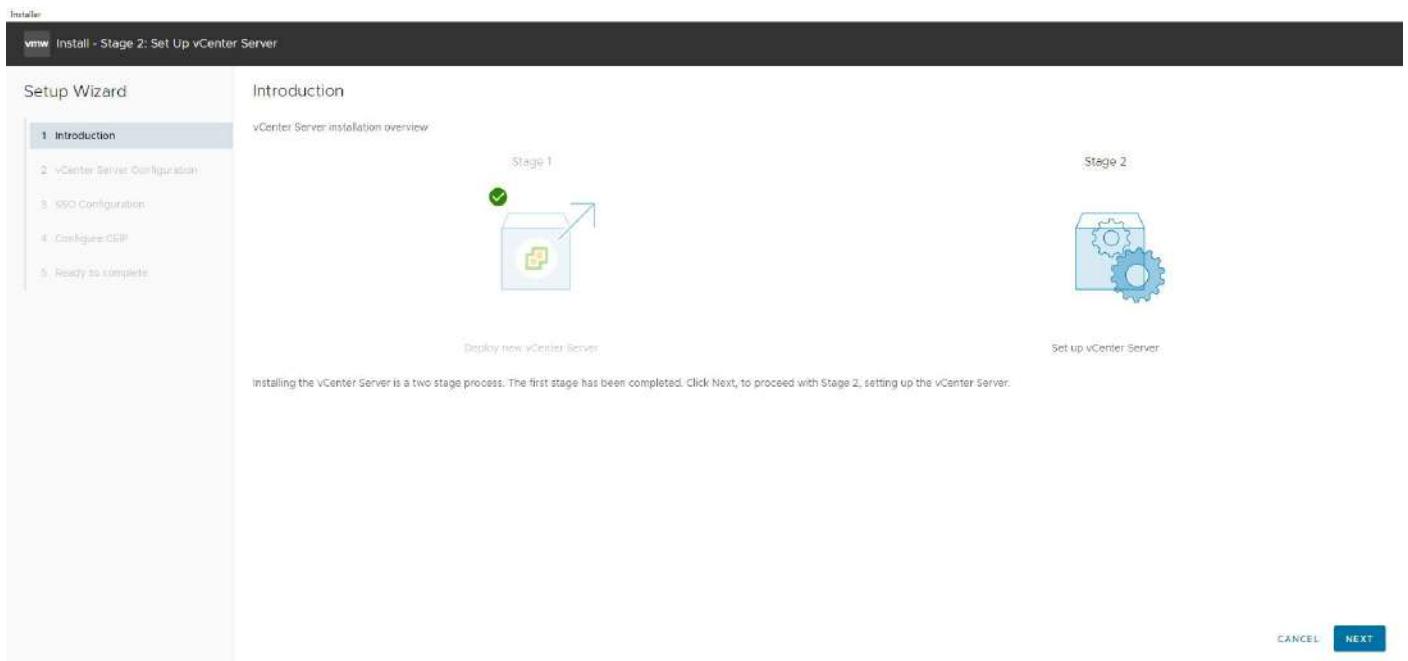
14. The vCenter Server will start deploying on the specified target ESXi host. Installation progress can be viewed on the screen.



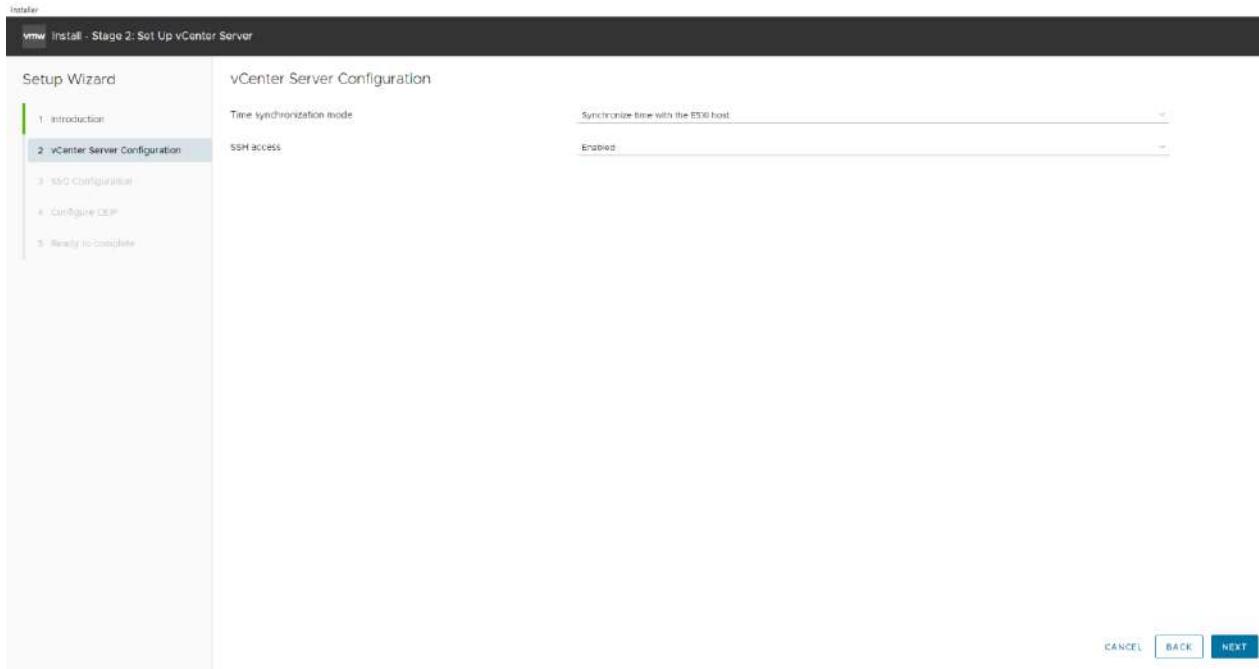
15. With the VCSA now deployed, move on to stage 2 by clicking **Continue**.



16. Select **Next** to proceed with Stage 2, setting up the vCenter Server.



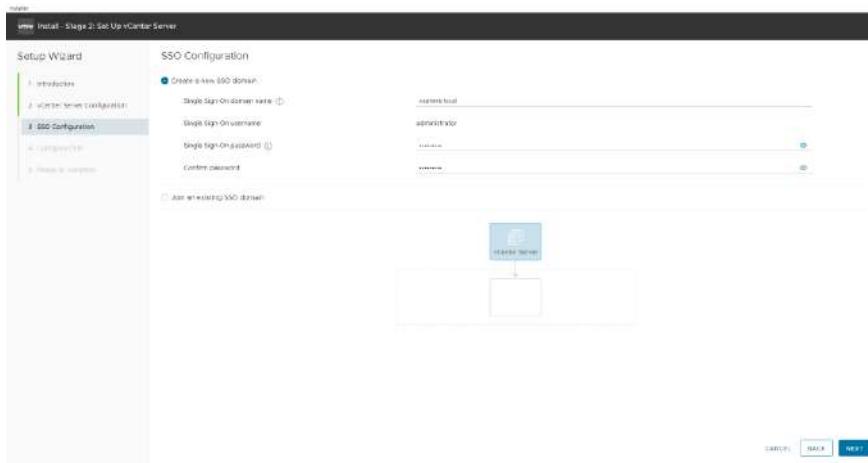
17. Configure the NTP server by selecting the Time synchronization mode and Enabling the SSH access, then click **Next**.



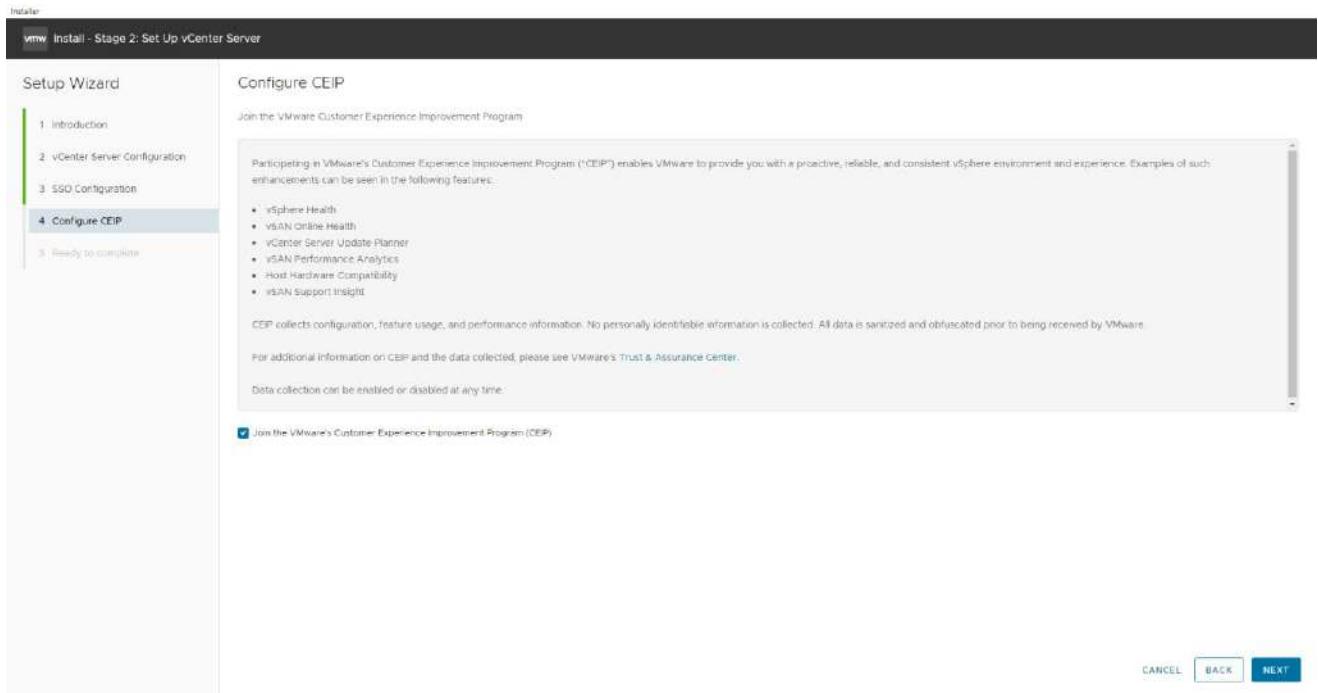
18. Enter a unique SSO domain name, configure a password for the SSO administrator, click **Next**.

Note

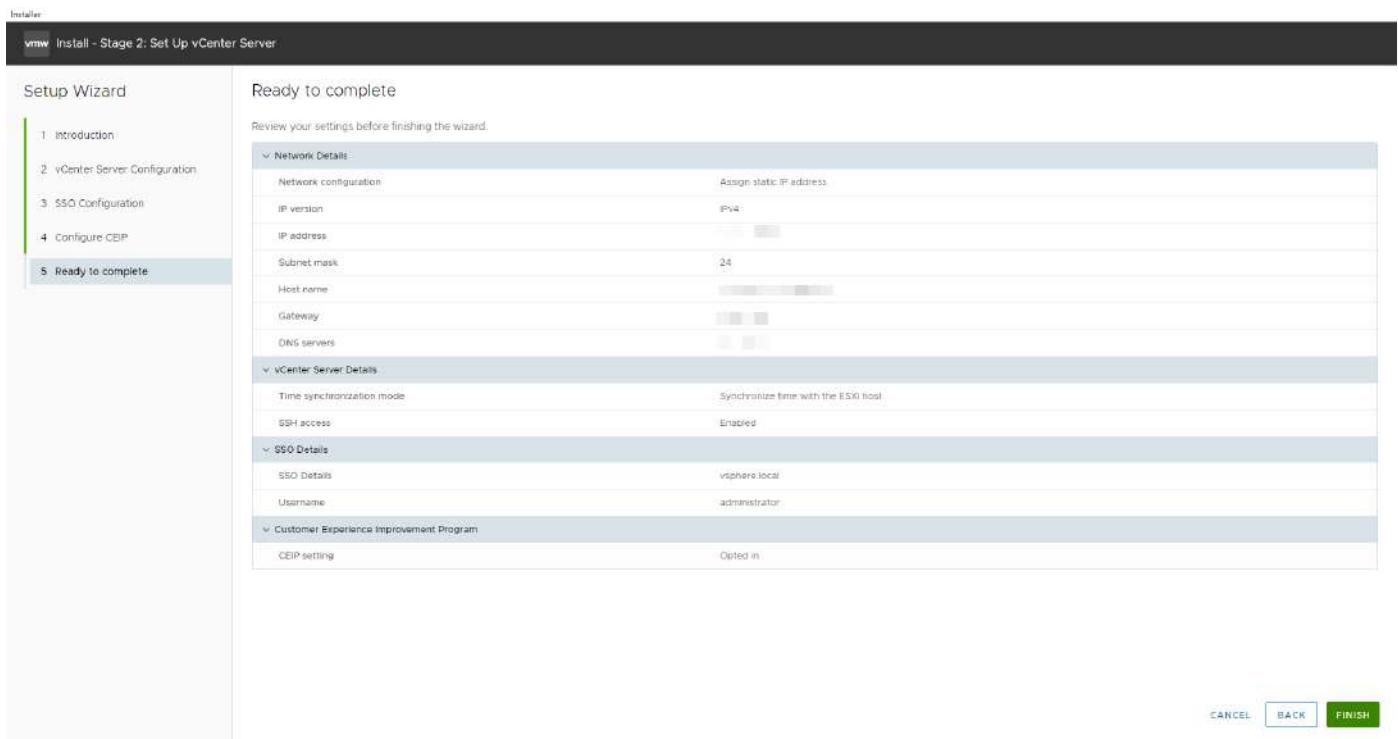
The default SSO domain name is vSphere.local. **The SSO domain name should not be the same as your Active Directory Domain.**



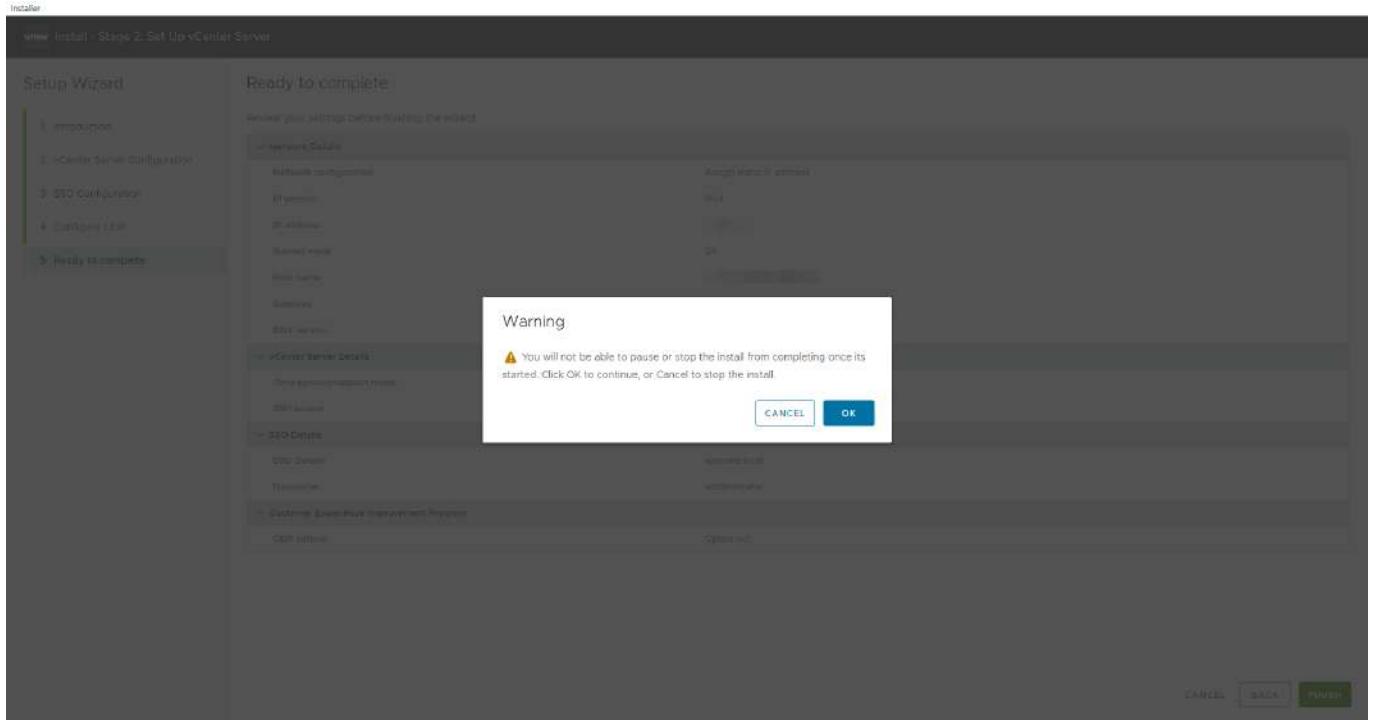
19. Select or deselect the customer experience improvement program box and click **Next**.



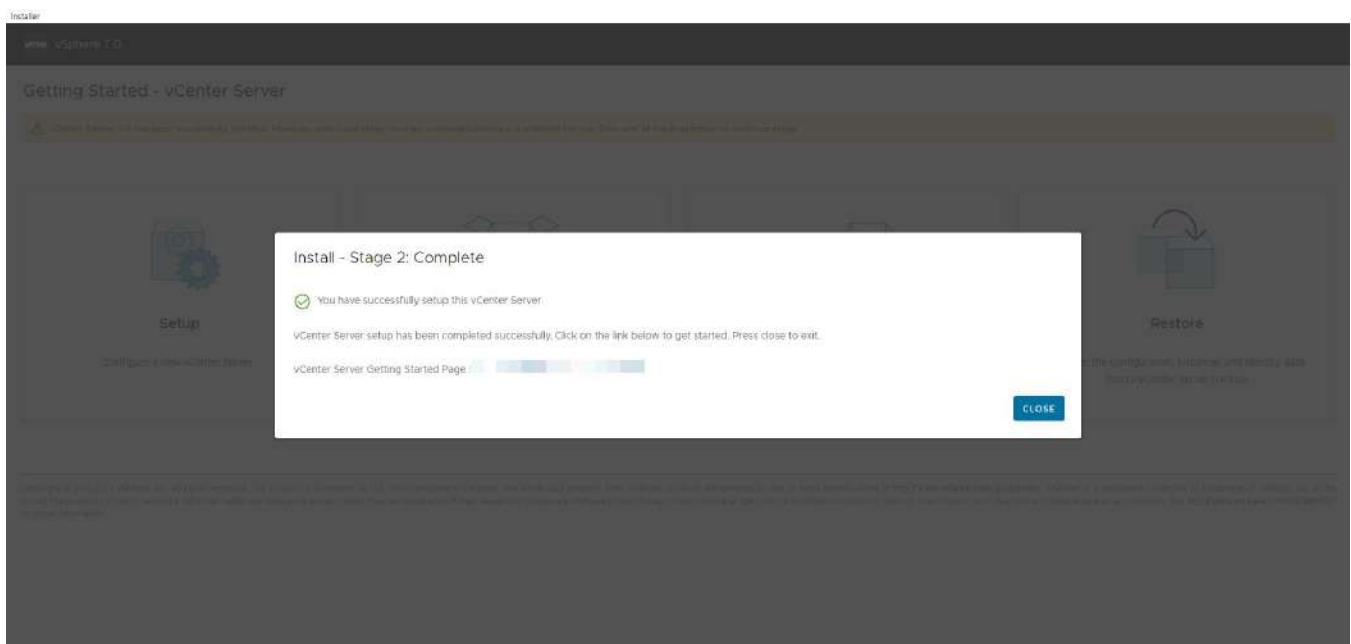
20. Review the details on the summary page and click **Finish** to finalize the setup.



21. The installer displays a warning that you cannot pause or stop the install once you start it. Click **Ok** to acknowledge the warning and start the install.



22. When the install process is complete, click **Close** to exit the installer and entire Stage 2 of the VCSA setup.



## Post Installation

This section describes the post-install and configuration of the vCenter Server.

### Adding Licenses to the vCenter Server

Use the following procedure to configure vCenter:

1. Connect to the vCenter post install using the IP or FQDN of the vCenter. Access vSphere by selecting **Launch vSphere Client (HTML5)**.

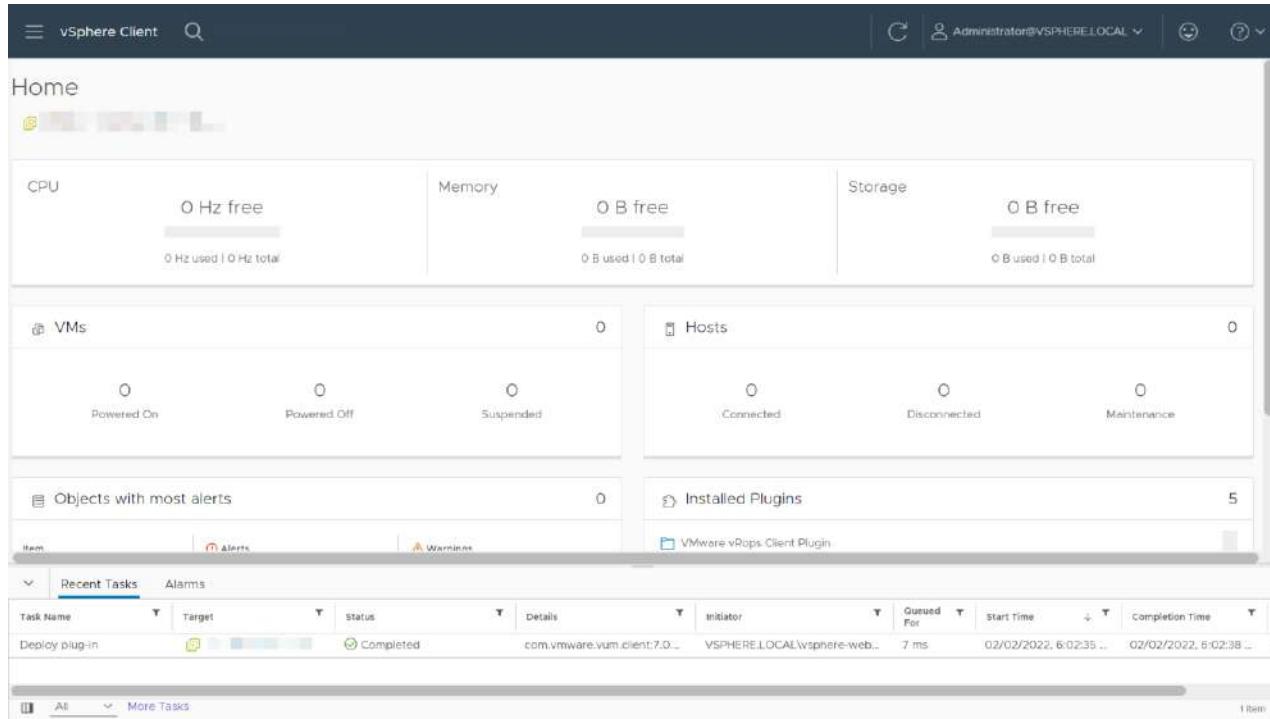
The screenshot shows the VMware vSphere Documentation Center. At the top left is the VMware logo. Below it is a navigation bar with 'Getting Started' and 'Documentation' tabs. A large blue button labeled 'LAUNCH VSPPHERE CLIENT (HTML5)' is centered on the page. To the right, there are sections for 'For Administrators' (Web-Based Datastore Browser, Browse datastores in the vSphere Inventory) and 'For Developers' (vSphere Web Services SDK, Learn more about the Web Services SDK, Browse objects managed by vSphere, Browse vSphere REST APIs, Download trusted root CA certificates). At the bottom left, there is a copyright notice.

Copyright © 1998-2021 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at <http://www.vmware.com/go/patents>. VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies. VMware products may contain individual open source software components, each of which has its own copyright and applicable license conditions. Please visit <http://www.vmware.com/info?id=1127> for more information.

2. The VMware Single Single-On page displays. Enter the username and password that you specified during installation, then click the **Login** button.



3. The VMware vSphere Web Client page displays.



4. You must apply for a new vCenter license key within 60 days. If you have purchased vCenter Server, log in to your [licensing portal](#). Select your license and log in to the vSphere Web Client using the SSO administrator login. (If the license key does not appear, then check with your VMware account manager.)
5. Click the **Menu drop-down**, then click **Administration**. Select Licenses from the left-hand menu, then select the Licenses tab. Click **Add New Licenses** to open the New Licenses dialog.

The screenshot shows the vSphere Client interface with the 'Licenses' tab selected in the navigation pane. The main area displays a table of licenses. A single row is present in the table, representing an 'Evaluation License'. The columns include License, License Key, Product, Usage, Capacity, State, Expiration, My VMware Notes, and My VMware Custom Label. The 'Evaluation' status is indicated by a yellow warning icon. At the bottom of the table, there is an 'EXPORT' button.

License	License Key	Product	Usage	Capacity	State	Expiration	My VMware Notes	My VMware Custom Label
Evaluation License			--	--	Assigned	Evaluation	--	--

6. Enter the vCenter Server Standard license key provided at the vSphere Licensing Portal.

7. Enter a unique name for the license in the License Name field and then click **Next**.

New Licenses

1 Enter license keys

**2 Edit license names**

3 Ready to complete

Edit license names

X

License name: License 1

License key: 

Product: VMware vSphere 6 Desktop Host

Expires: Never

Capacity: 100 VMs

CANCEL BACK NEXT

8. Review your selections, then click **Finish** to close the Enter New License dialog and return to the VMware vSphere Web Client page.

New Licenses

1 Enter license keys

2 Edit license names

**3 Ready to complete**

Ready to complete

X

Number of licenses: 1

License name: License 1

License key: 

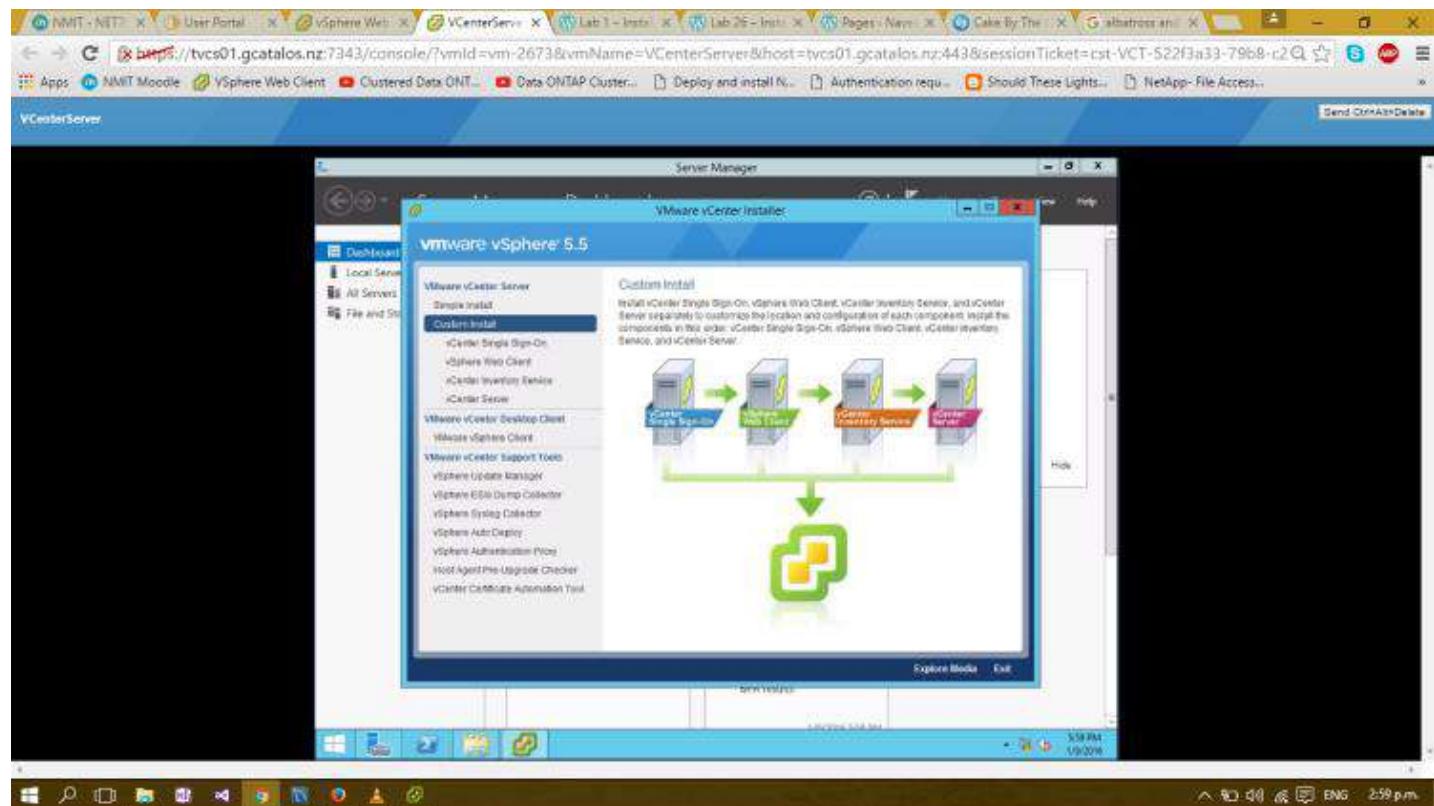
CANCEL BACK FINISH

## 2. Configure Single Sign-On and Create a Data Center Object.

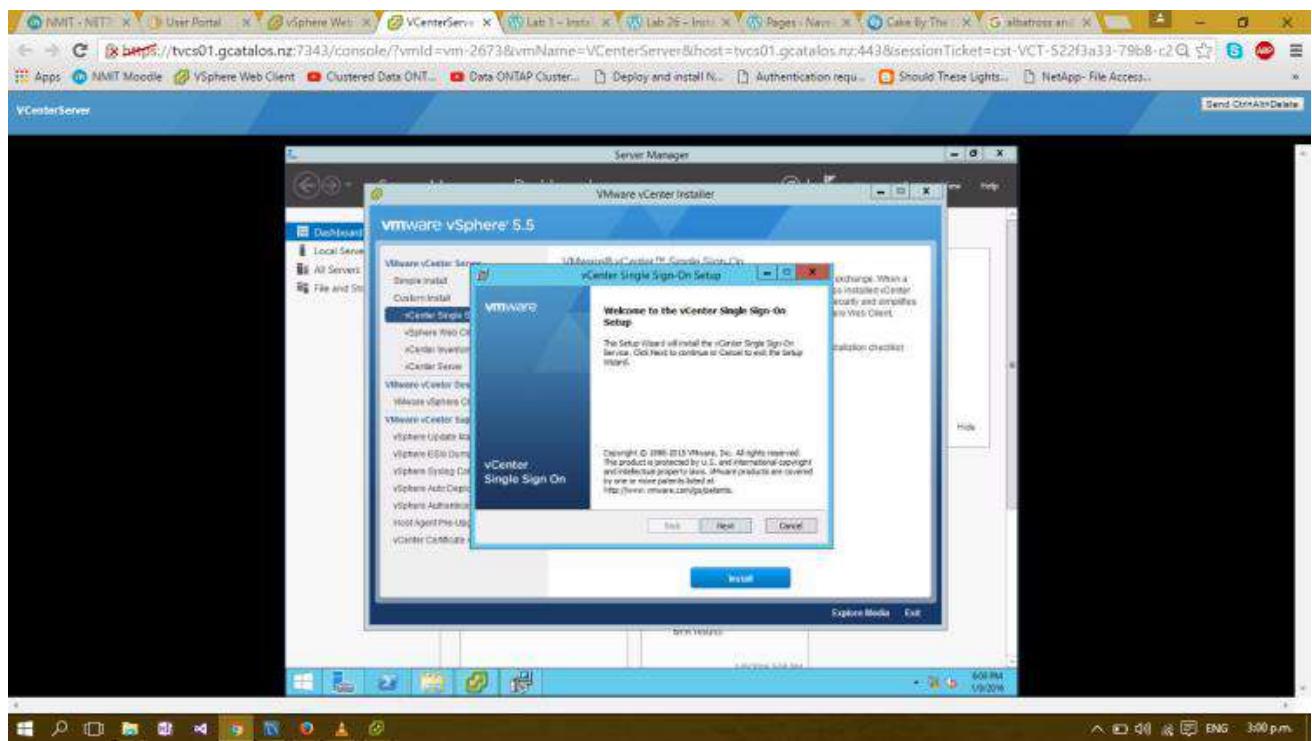
### Installing Vcenter Single Sign-On

So, Lets start with installing the Vcenter Single Sign on Server. Click on Run you can see a pop up window showing the installing services on the left pane.

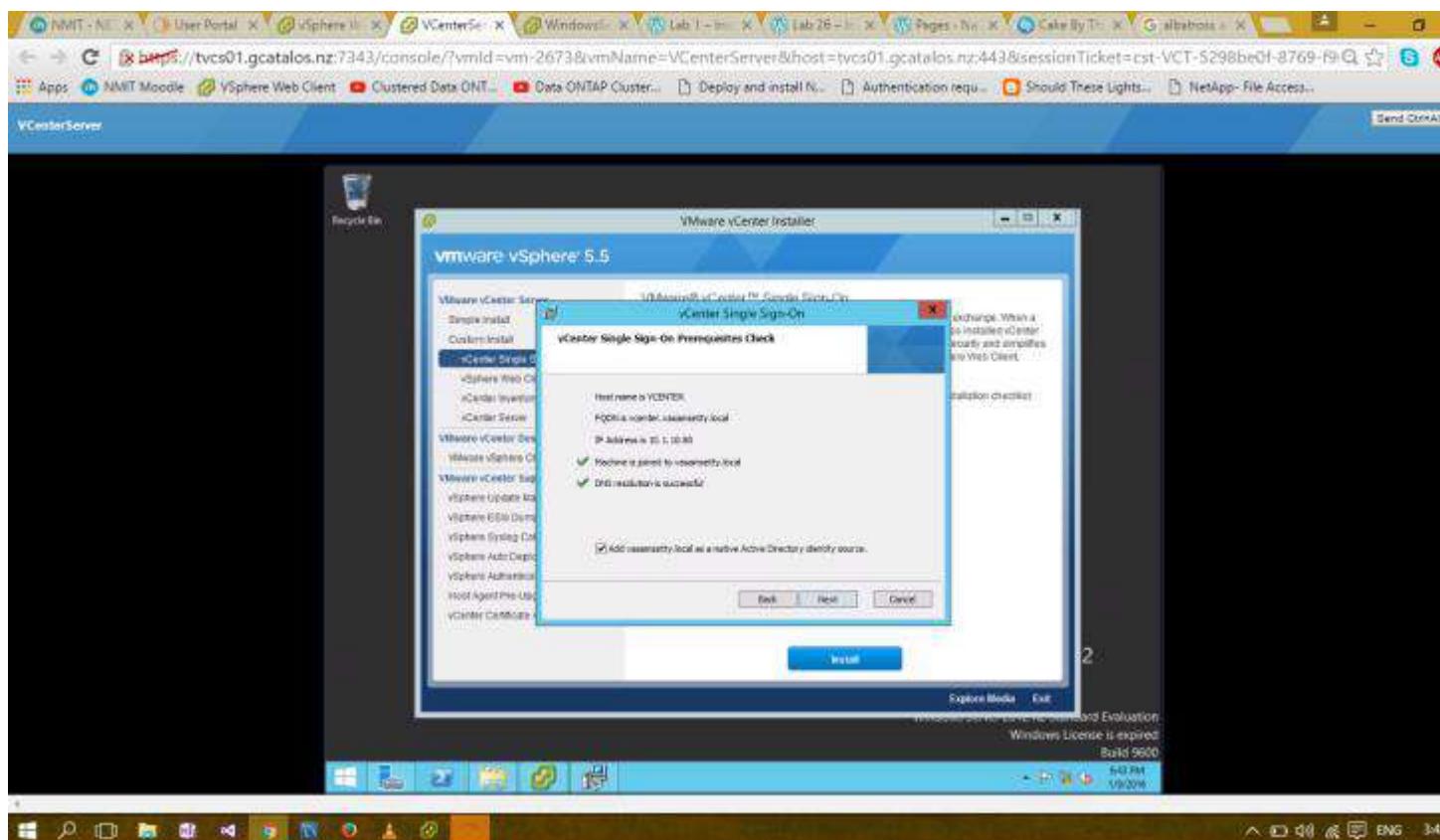
There are two options of installation process you can choose either a simple install or a custom install. We go for custom install since we need to know the steps that are processed on installing of each service.



Click on **Single Sign-on > Install**

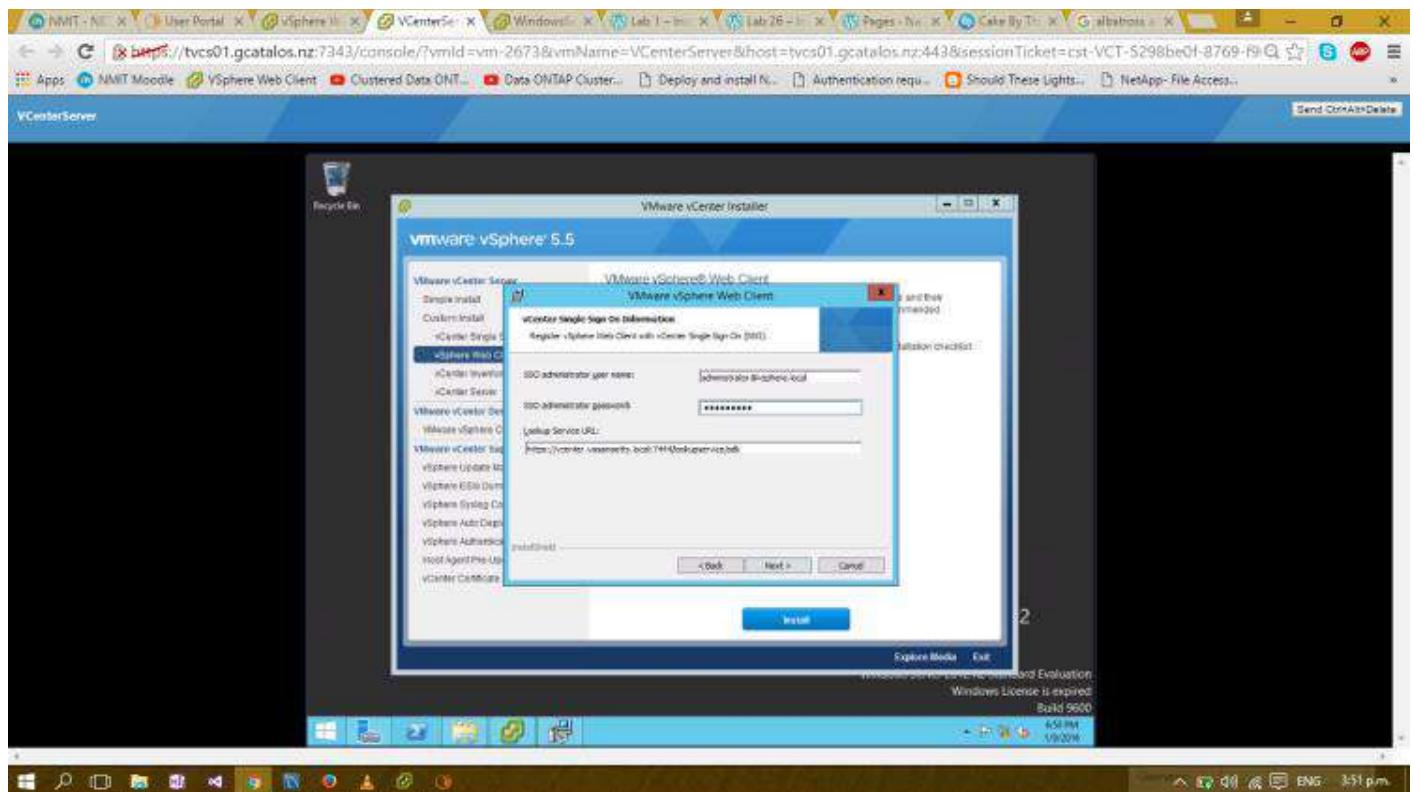


check the prerequisites had passed without any errors. hostname, FQDN, Static IP and DNS Resolution.



- Click on ‘Standalone vCenter Single Sign-On Server’
- Site name – Training-Site

We mostly had the default values on each step and moved on. Please make sure you given the master password which you remember the most.

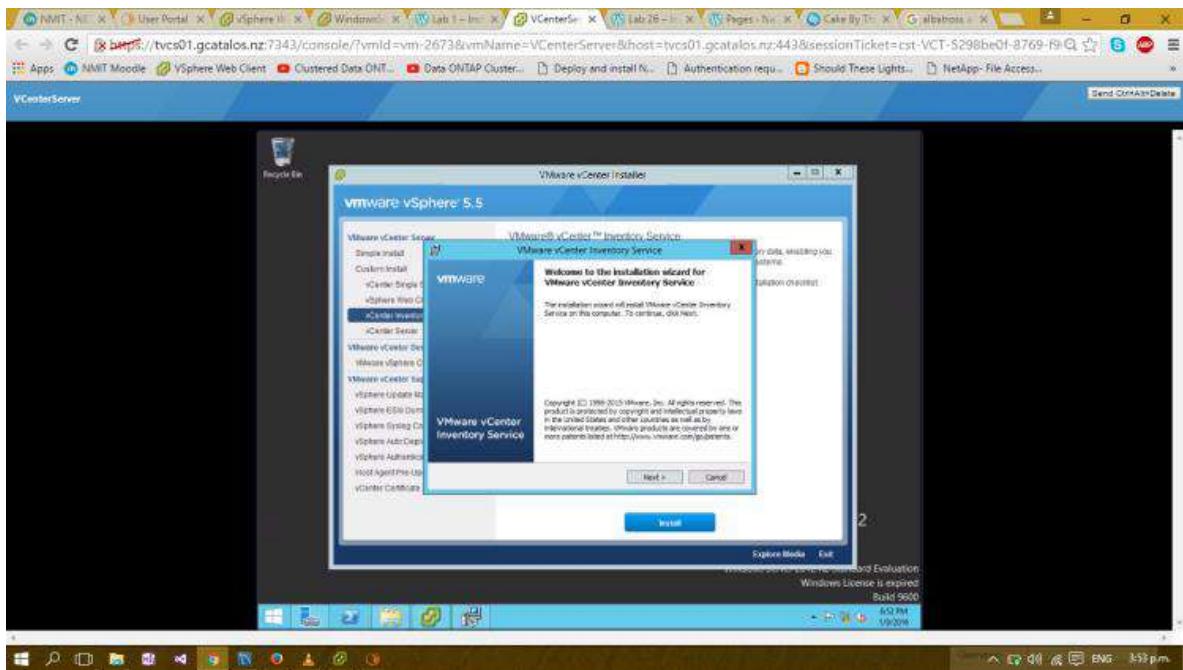


Proceed with the further steps to install the SSO, After the installation is done move to the next step installing Inventory Service.

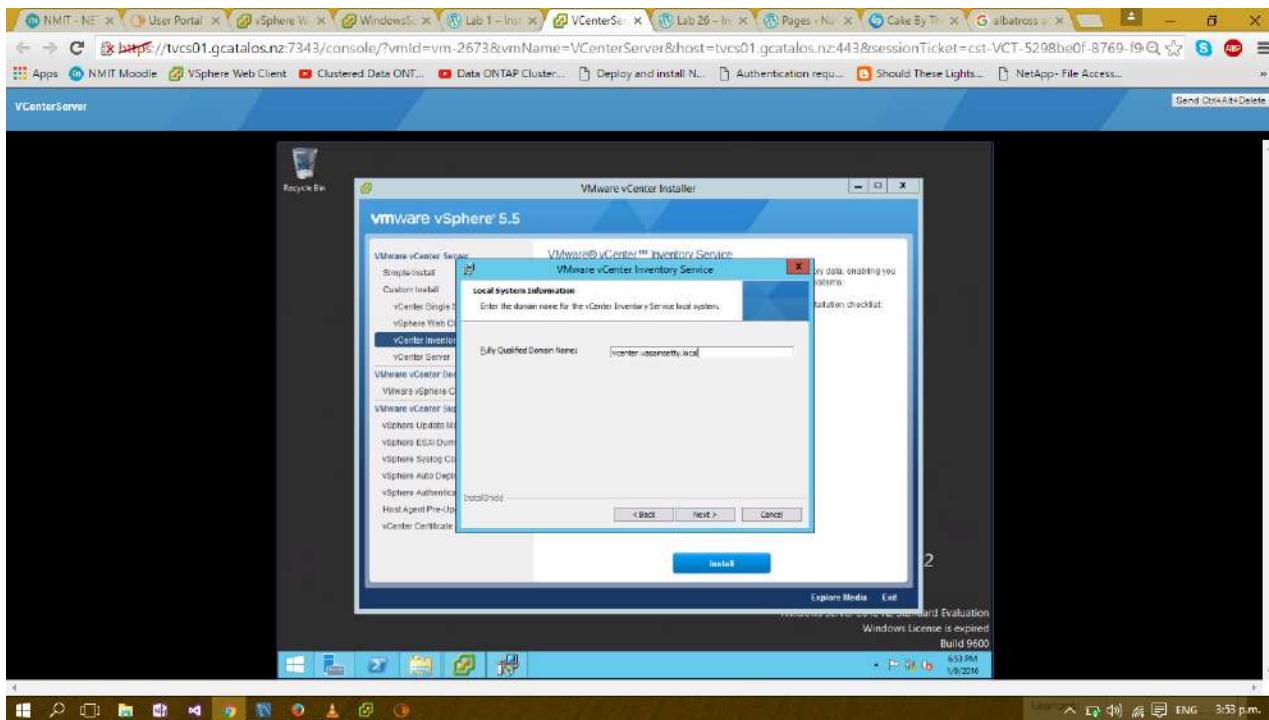
### **Installing Vcenter Inventory Service:**

In the second task we install vcenter inventory services,Inventory service stores vcenter server application and inventory data allowing you to search and access inventory objects across linked vCenter Server systems.

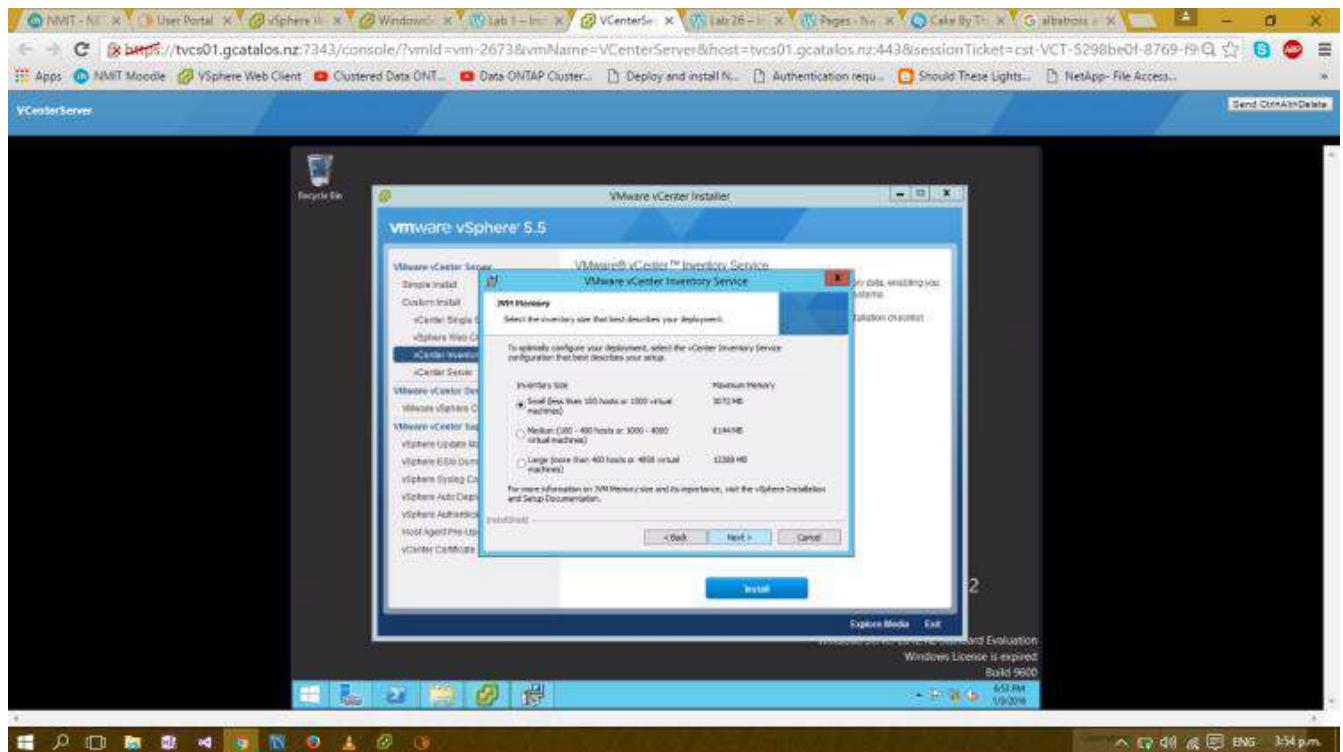
Goto the window that pop-up at the beginning of the lab.



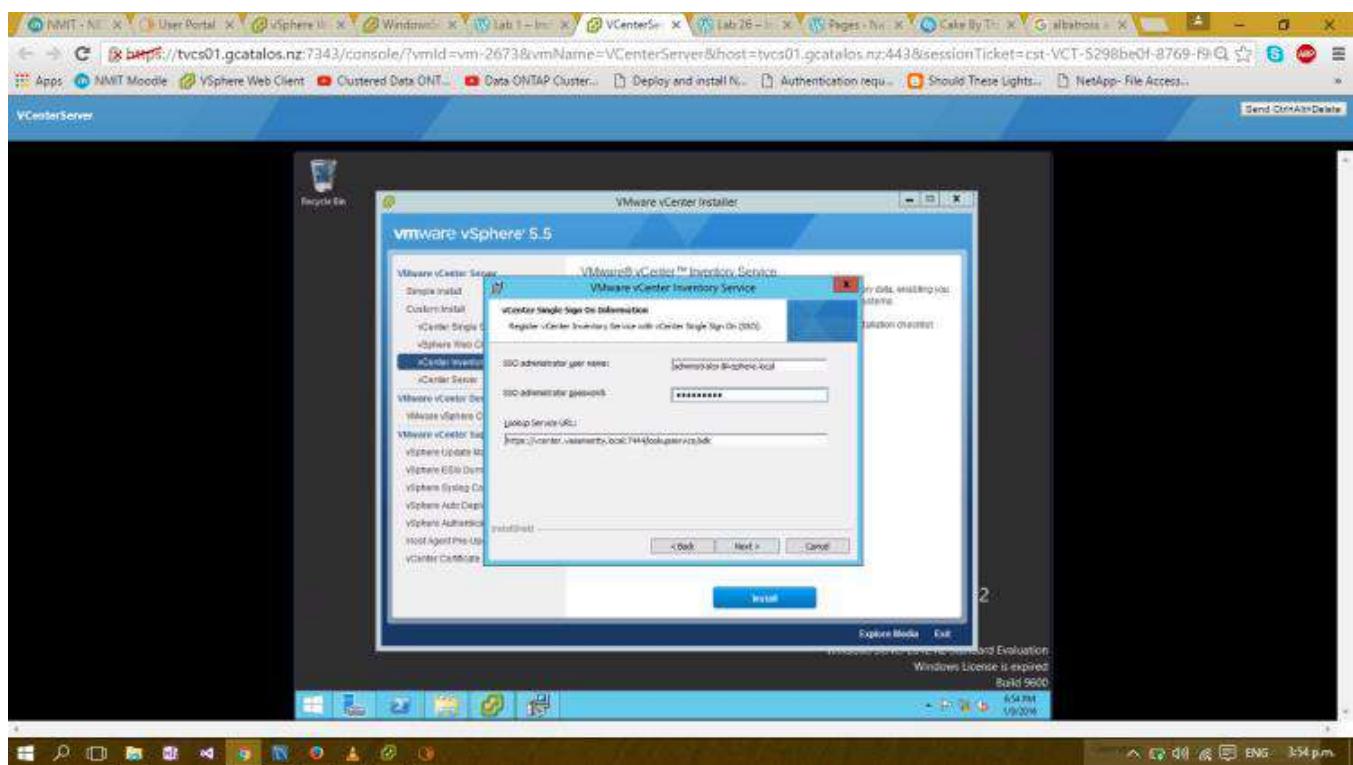
click on **Vcenter Inventory Service** and press **Next**. you are being asked to write the **FQDN (Fully Qualified Domain Name)** just enter your domain name and **next**. In the next window provide the default ports click on **Next**



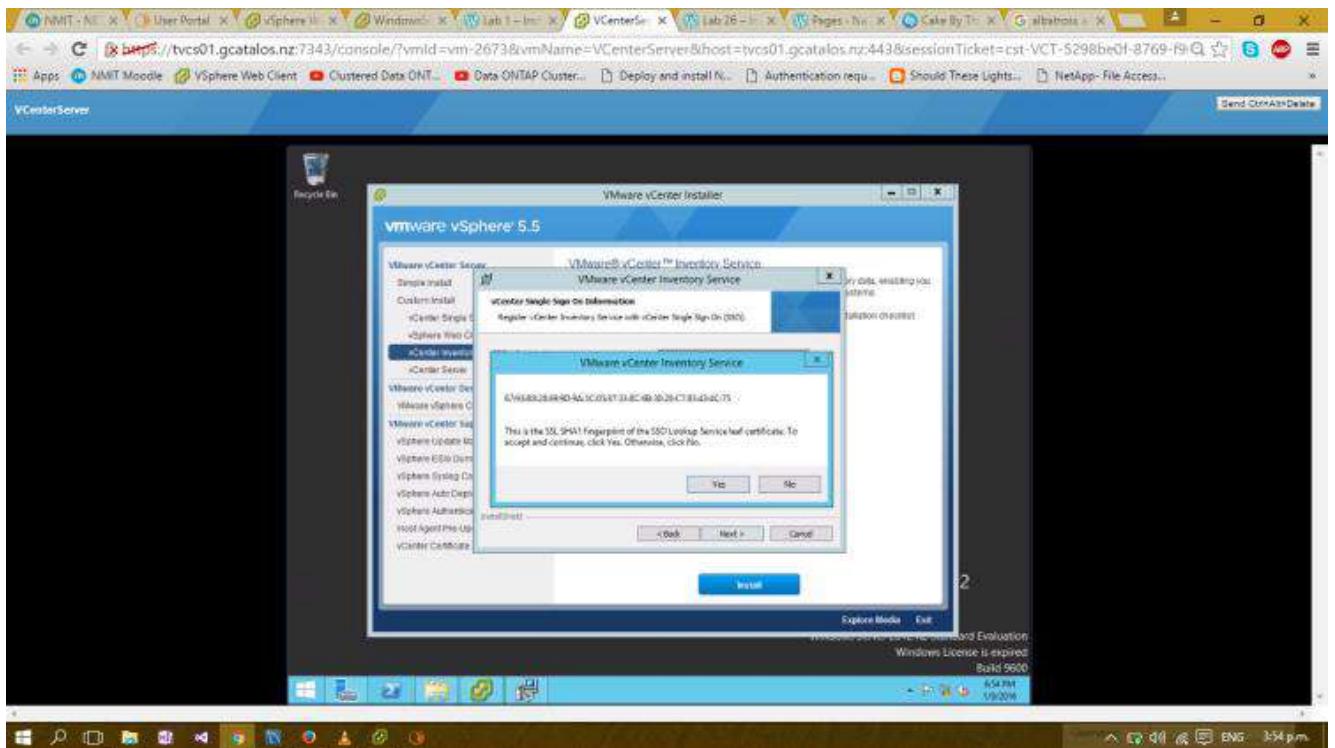
For the inventory size, I choose the smallest option available as we won't be creating many virtual machines at all for this duration of this lab.



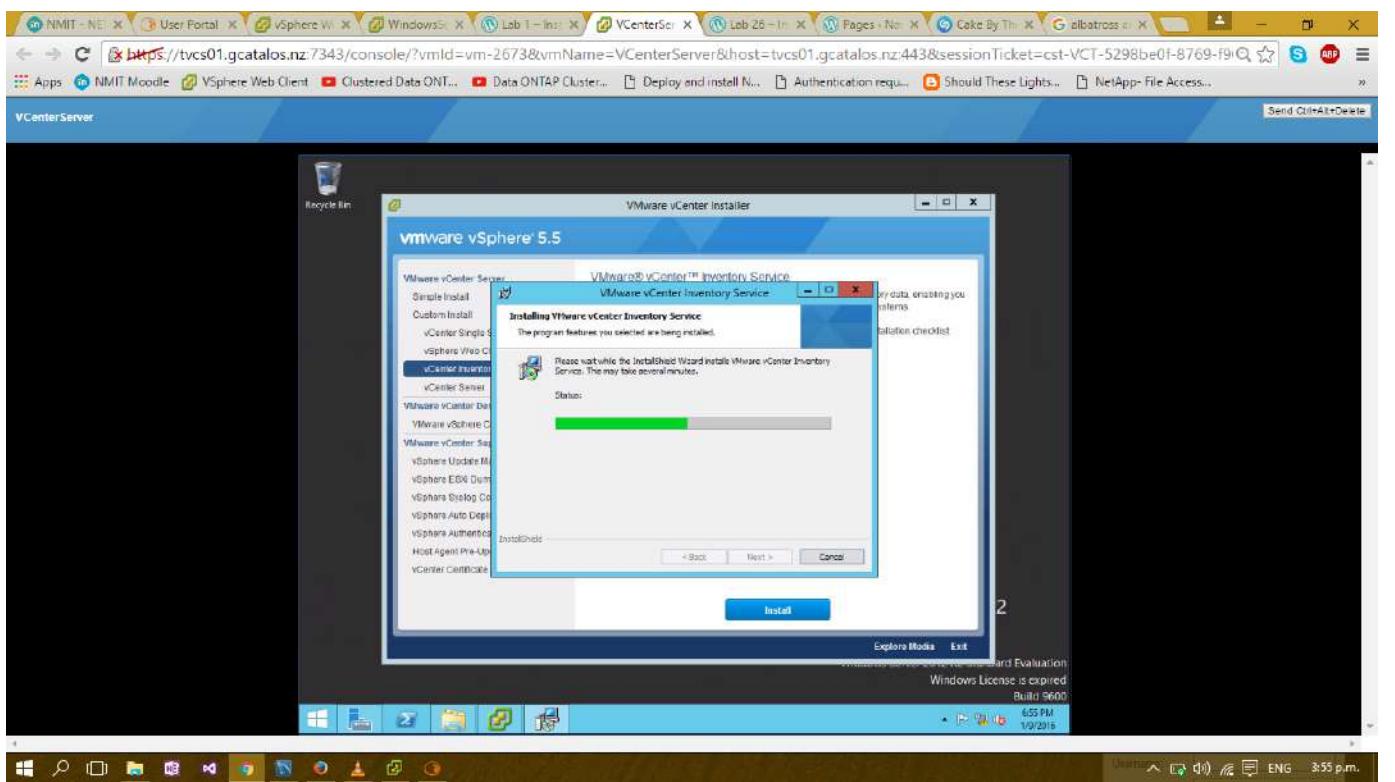
I then entered the password that I defined earlier during the Single Sign-On installation for the Administrator account.



This made me to agree and install the certificate.



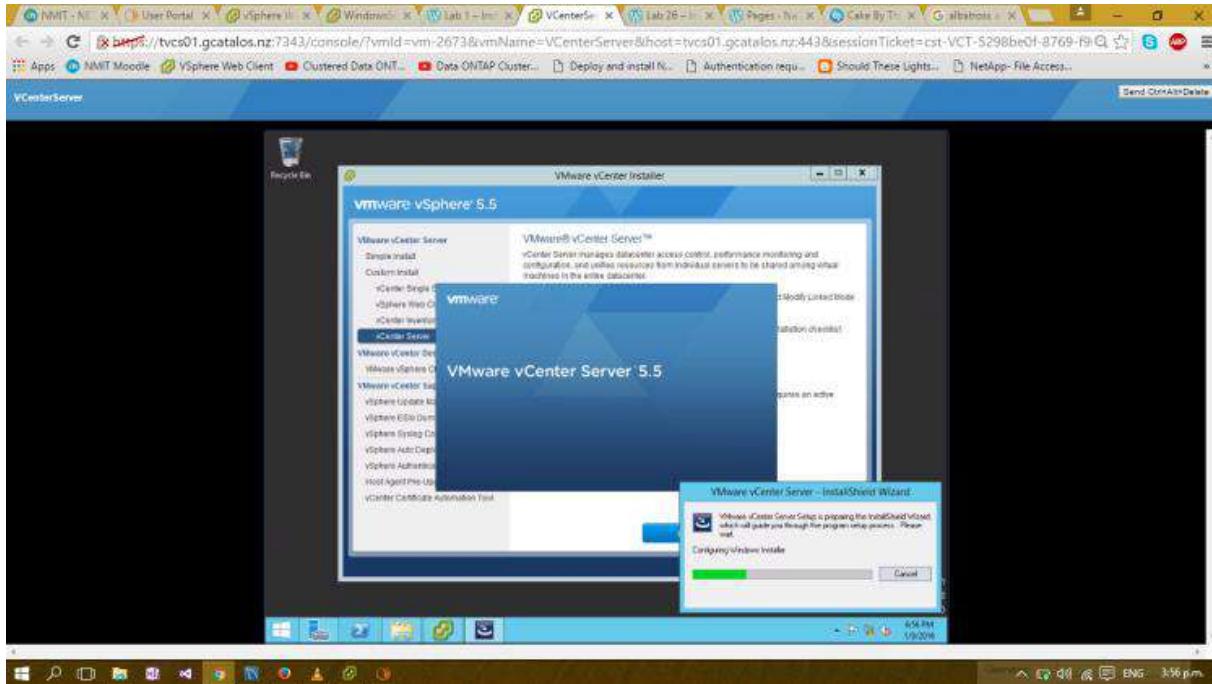
Once the certification is installed. Move on to next and Click **Install**



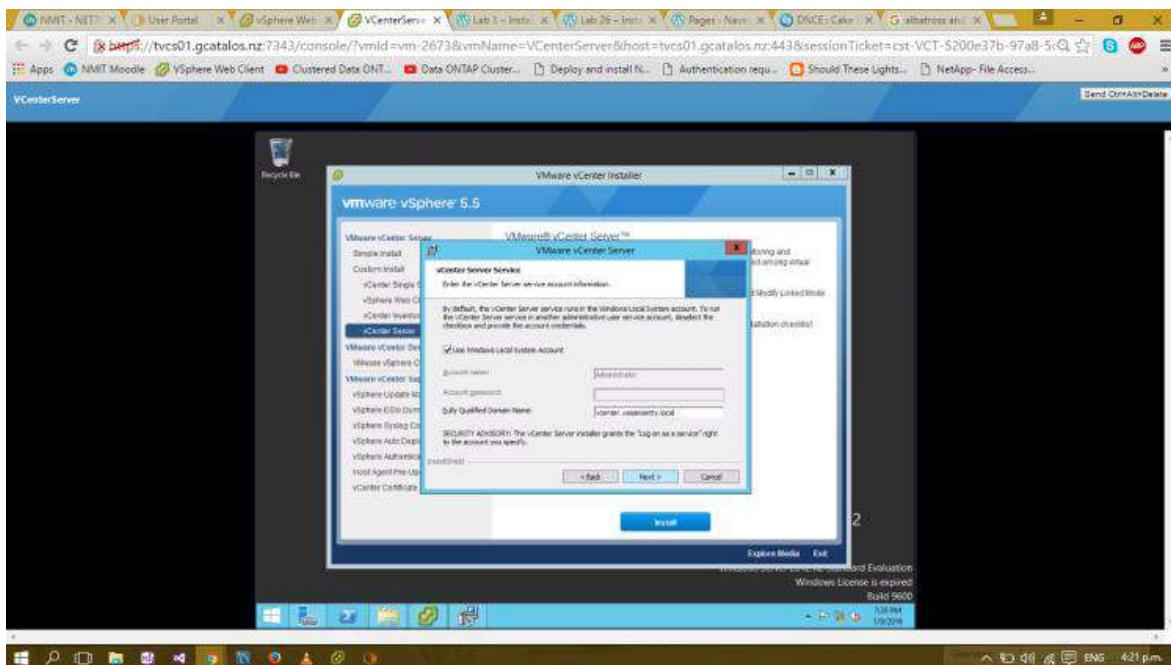
After the installation had completed, I moved on to installing the vCenter Server.

## Installing Vcenter server :

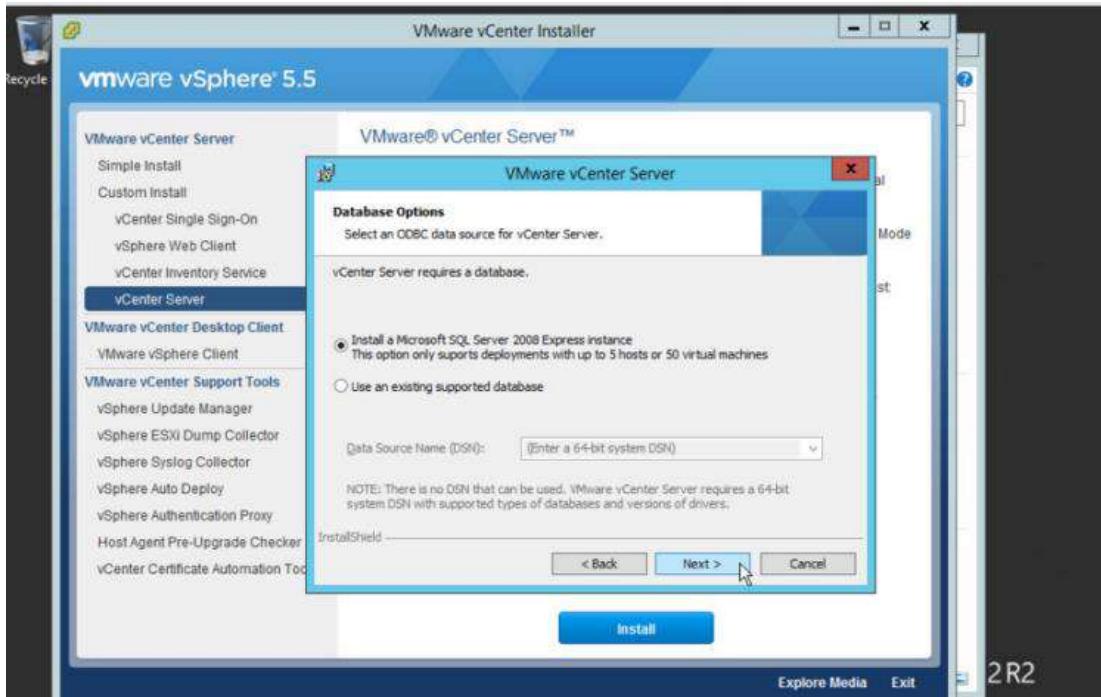
VMware vCenter Server allows for the management of multiple ESX servers and virtual machines (VMs) from different ESX servers through a single console application. Before installing the Vcenter Server, the application configure .Net Framework which i had enabled it earlier starting of the lab.



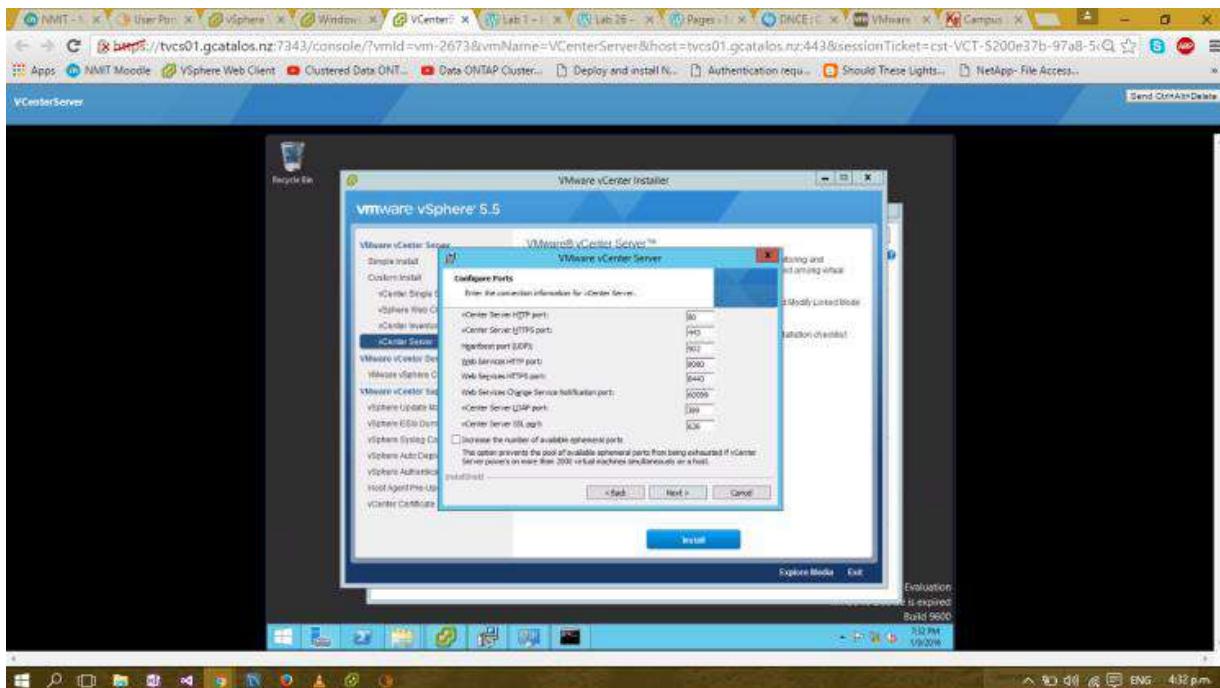
Enter your FQDN details

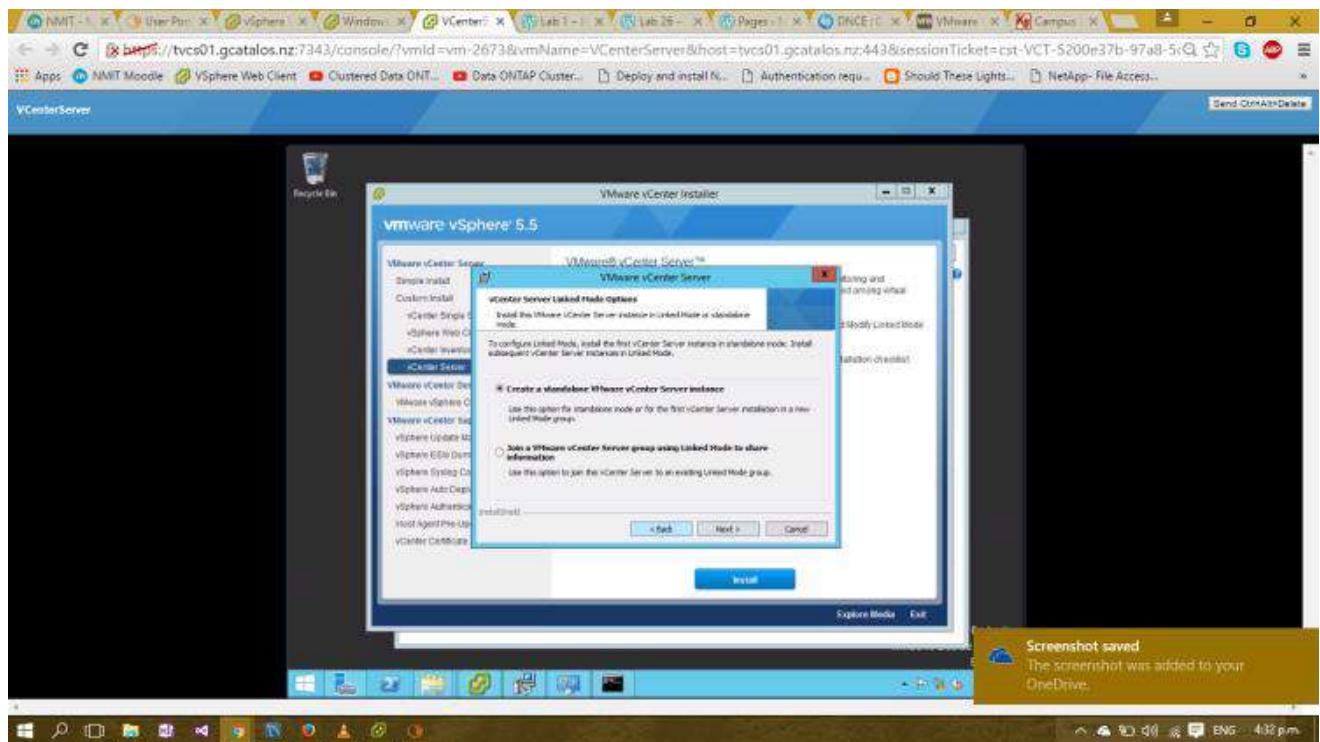


in the next step Vcenter server must install a database. Once we move from previous to next step , it asks to select a exist database or a new data source. I choose the option of installing a Microsoft SQL Server 2008 Express instance. The instance we created supports for our available hosts.

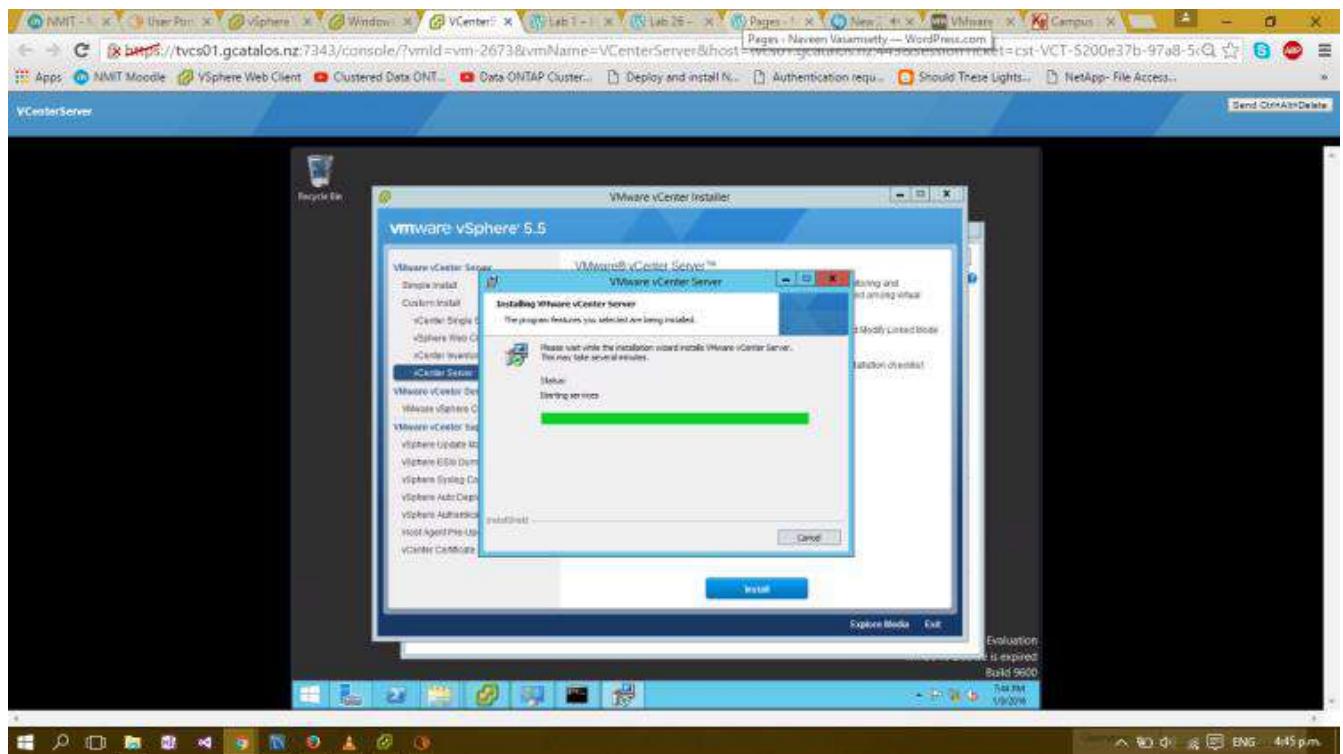


i Just moved on selecting the default values since it was the first installation.



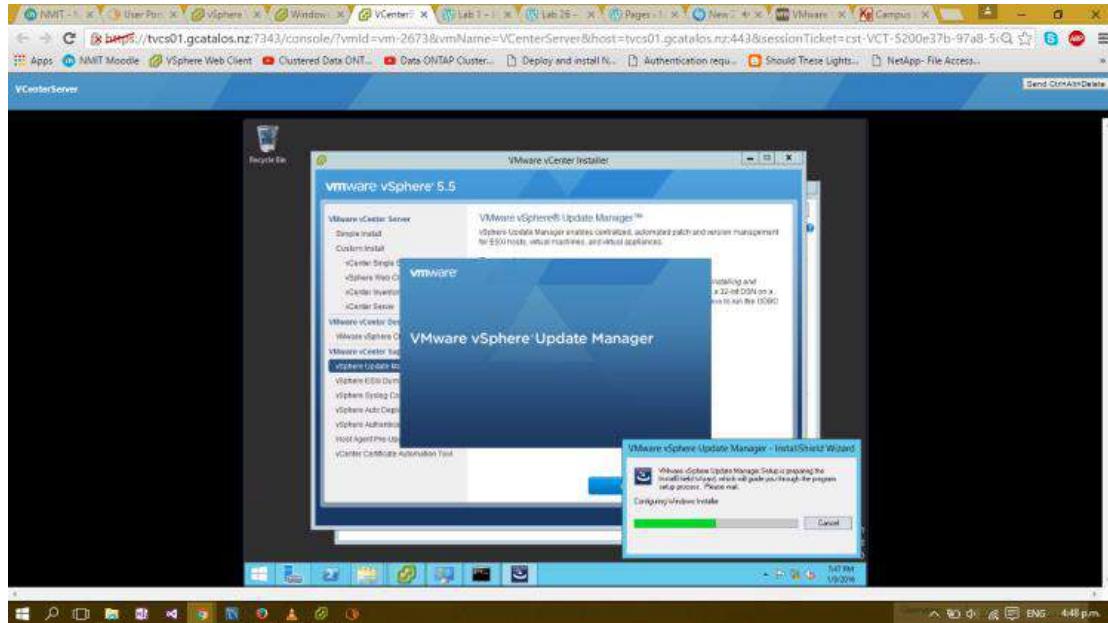


Once all the configuration is done we come at the install window.Click on install to begin the install.

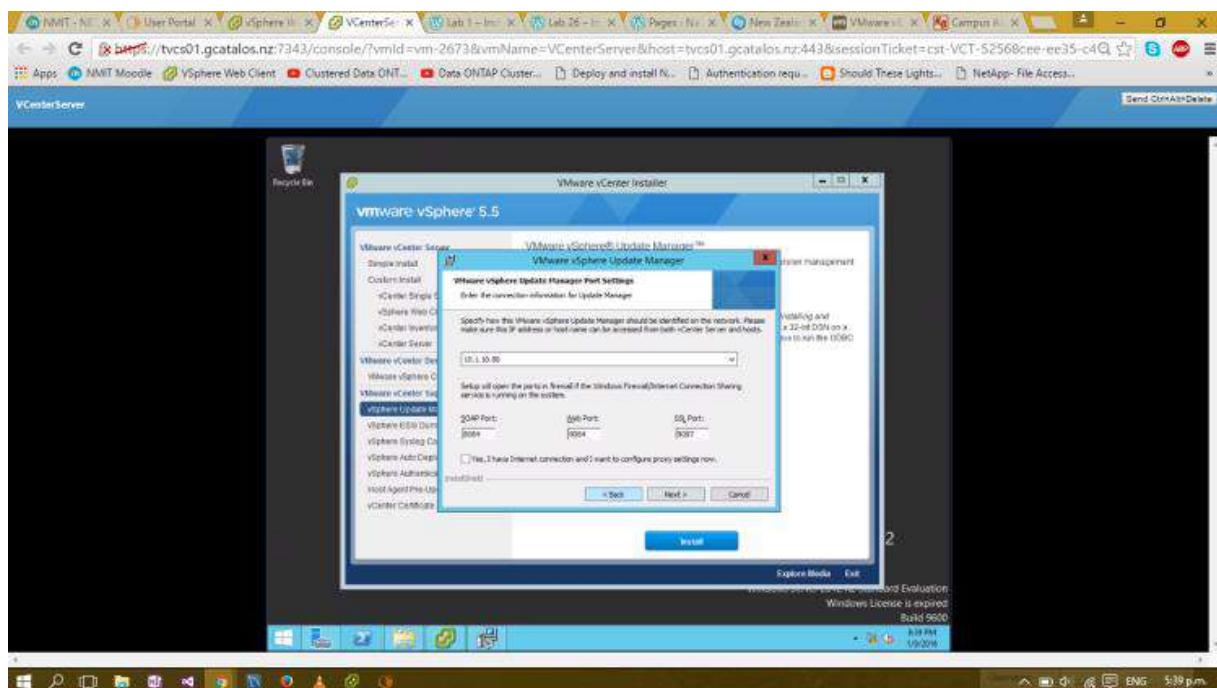


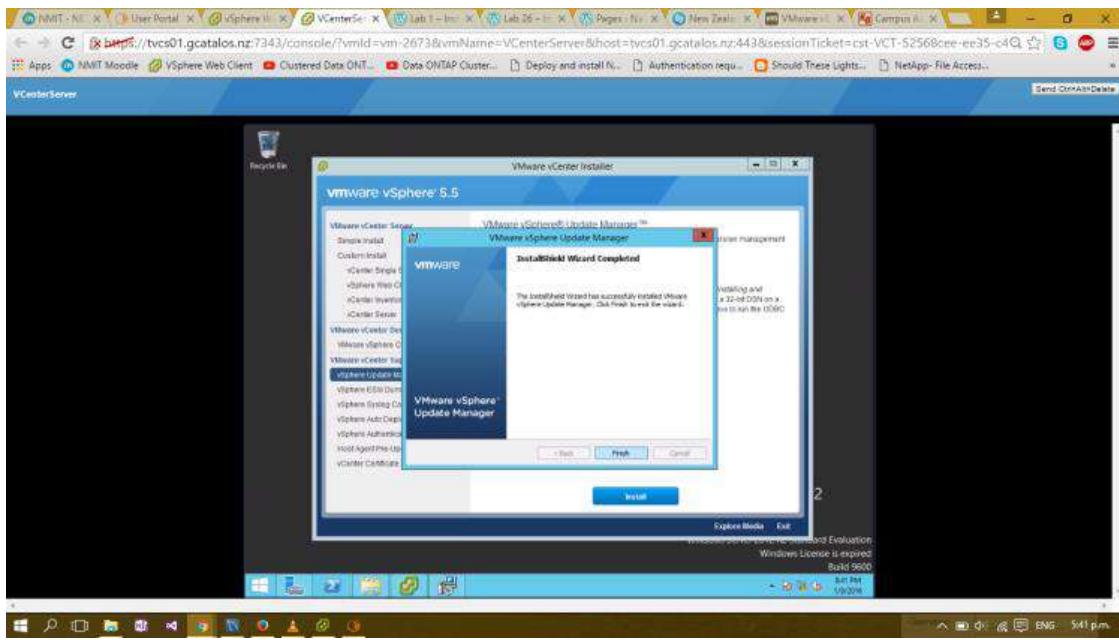
Now the final task installing VSphere update manager. In the update manager it enables centralised, automated patch and version management for ESXi hosts, virtual machines, and virtual appliances.

Install update manager by clicking install on the main pop-up window. In this installation process we leave the configuration options to default and move the process.



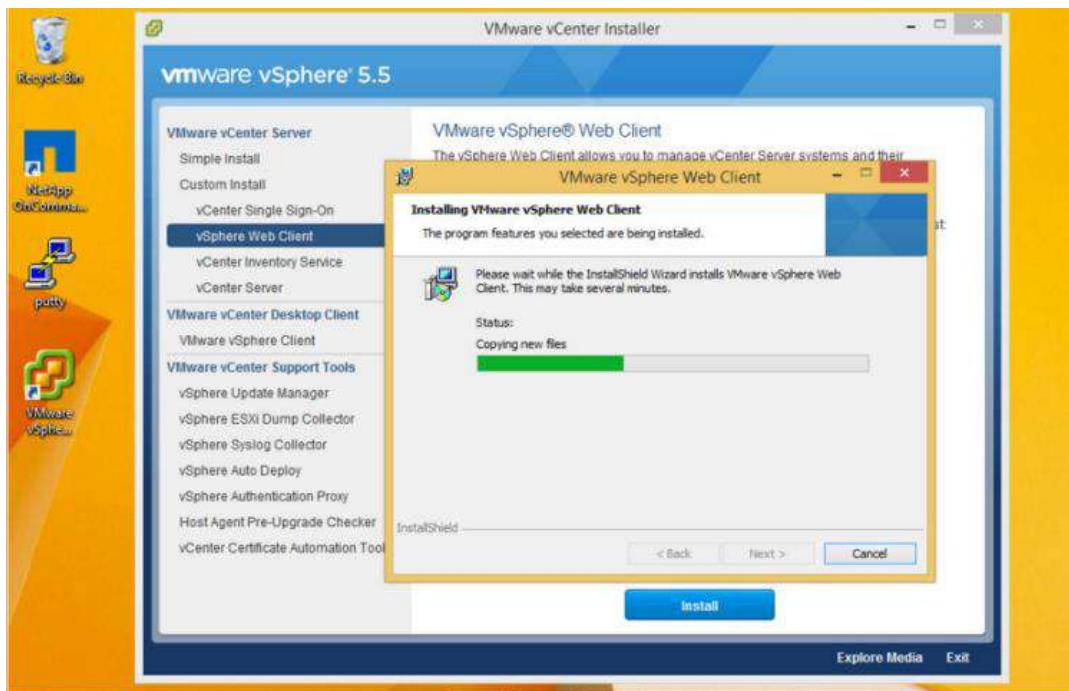
For the VMware vCenter Server Information, I provided credentials ad left the IP address and HTTP port as their default values.

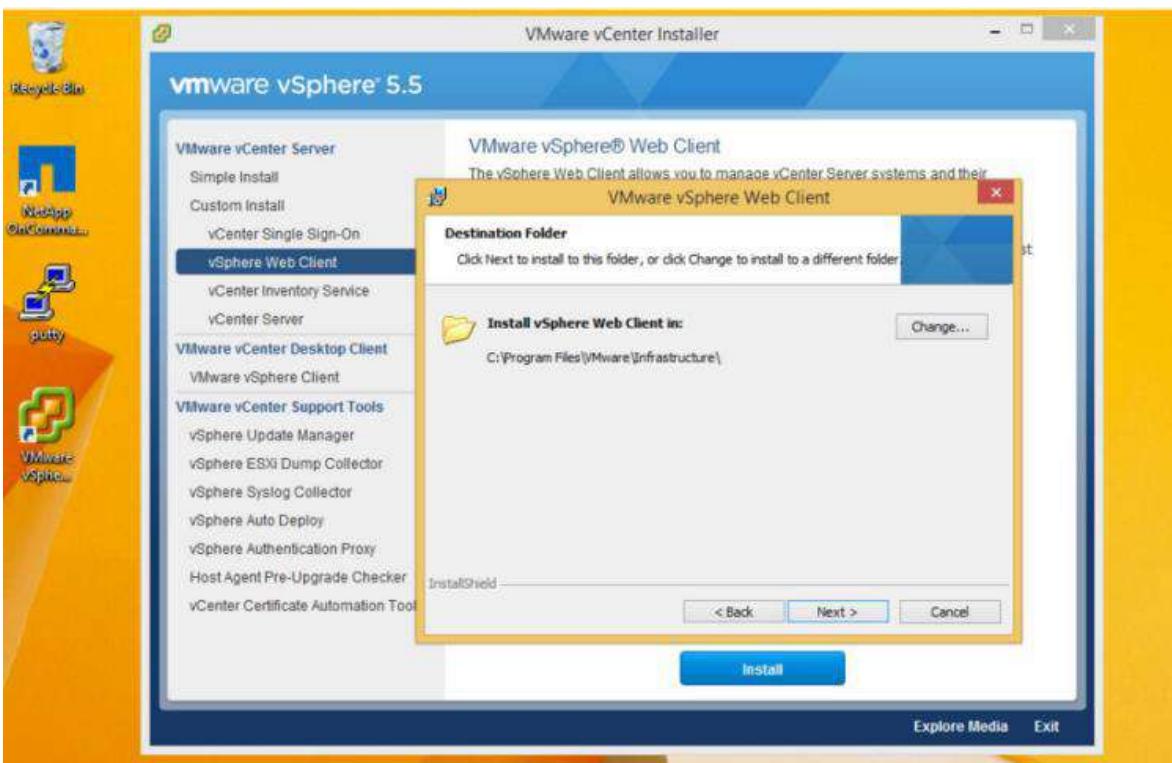
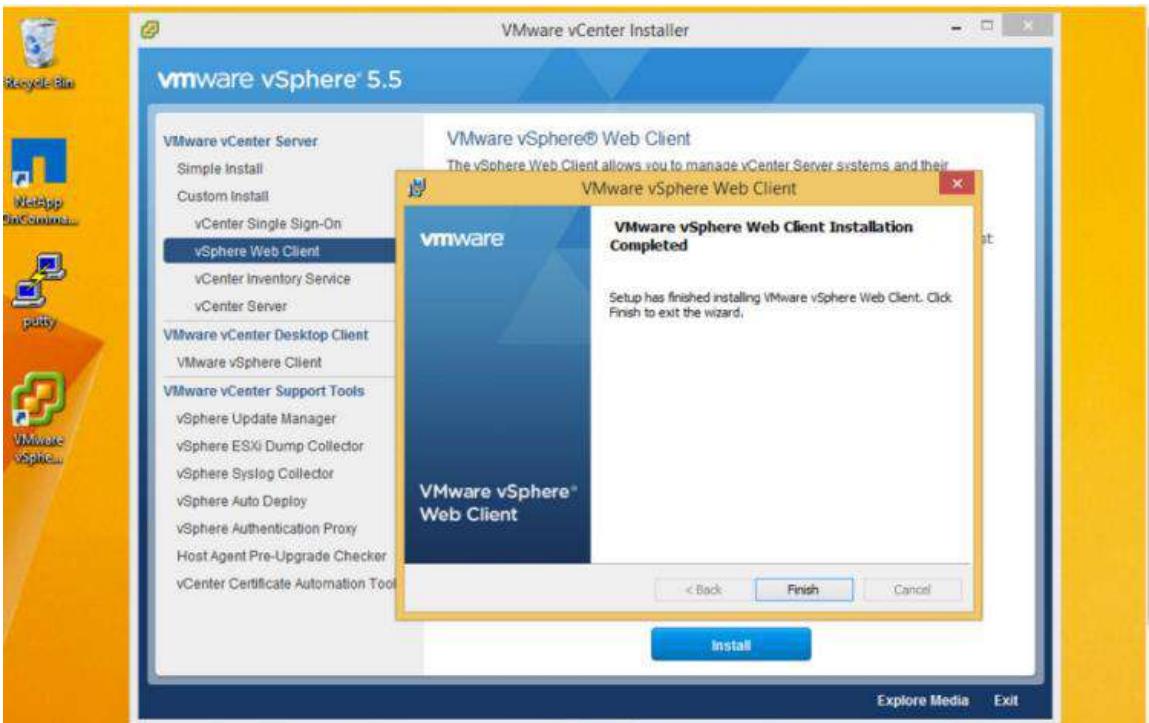




Finally we came up near to the end of the project once we finish installing the web client. May be we don't need to have a web client service on our vcenter server rather we install the web client on our windows client machines. I can say two points supporting my word. firstly as a security stand point and second we don't use much running service of web client on our vcenter server machine.

I just mounted my vcenter.ISO on my windows client and mounted it. Try to install the web client on it.

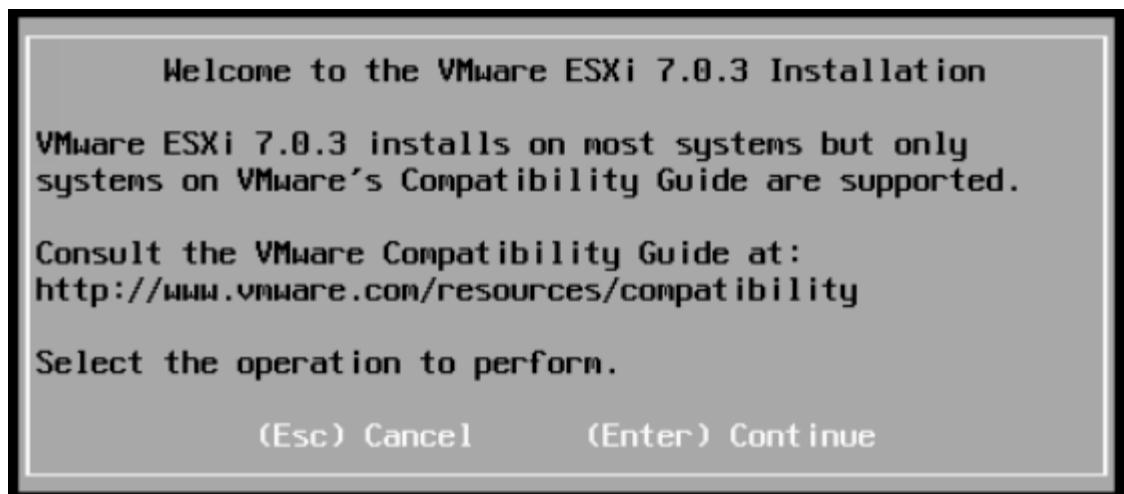




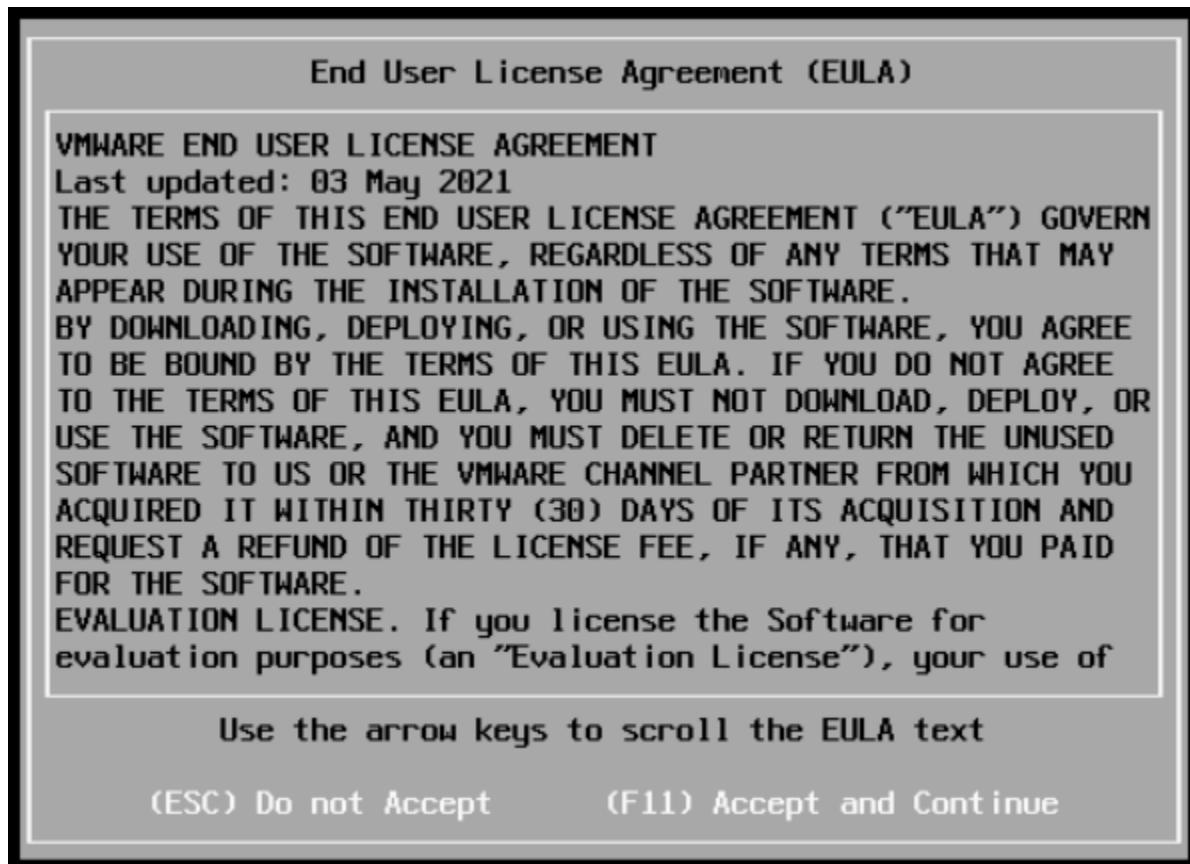
### 3. Add Your ESXi Hosts to the vCenter Server Inventory

Installing VMware ESXi

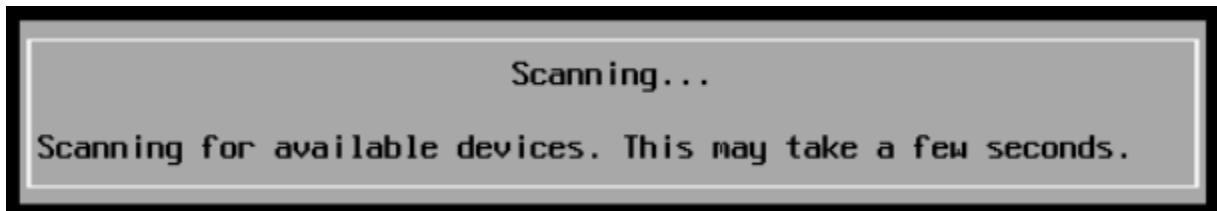
1. Apply power to start the host.
2. Select the installer using the arrow keys and press [ENTER] to begin booting the ESXi installer. A compatibility warning is displayed.



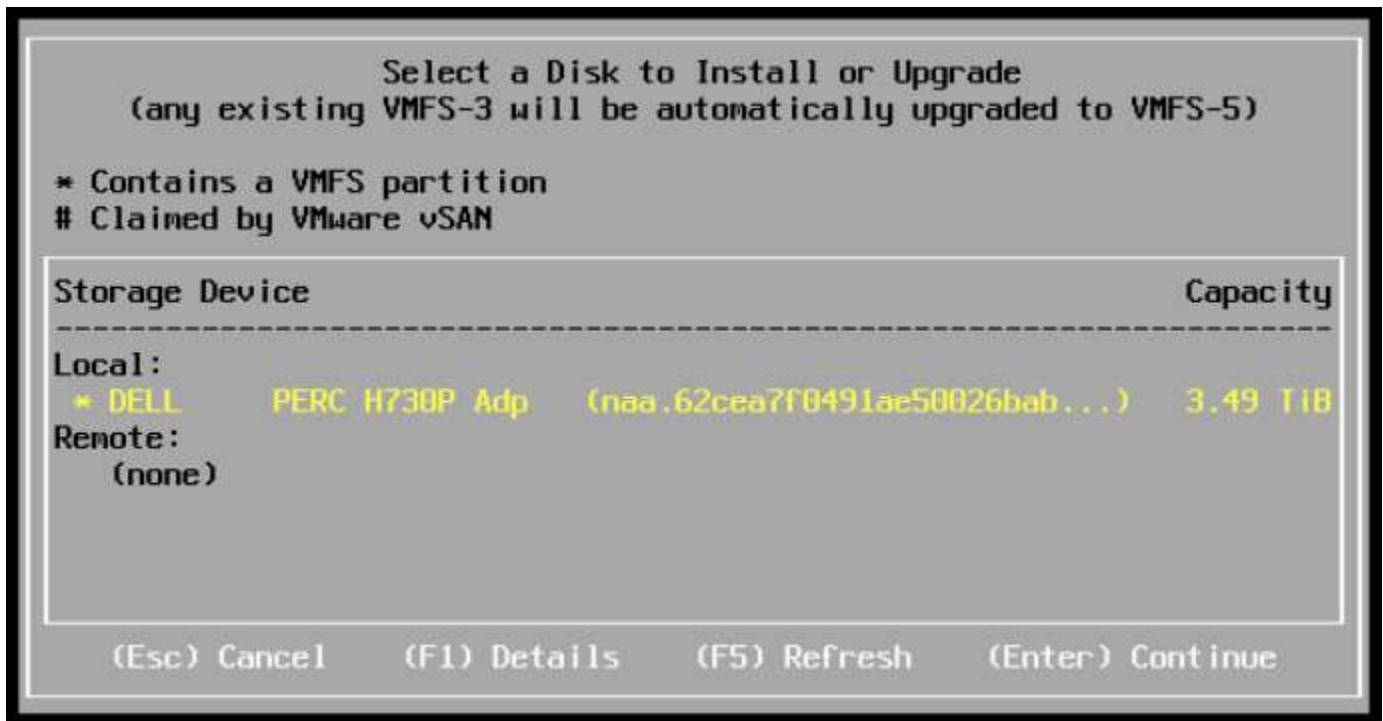
3. Press [ENTER] to proceed. The End User License Agreement (EULA) displays.



4. Read the EULA and then press [F11] to accept it and continue the installation. The installer scans the host to locate a suitable installation drive.



5. It should display all drives available for install.

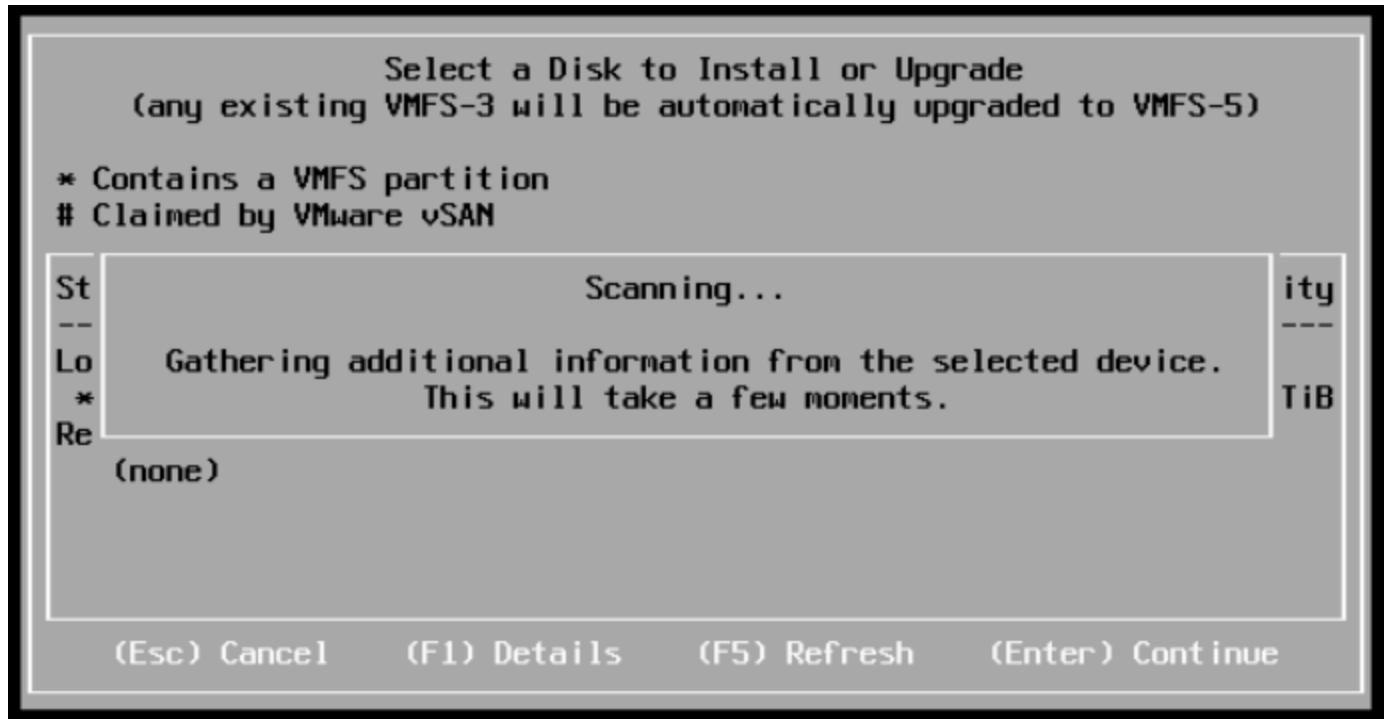


6. Use the arrow keys to select the drive you want to install ESXi, and then press [ENTER] to continue.

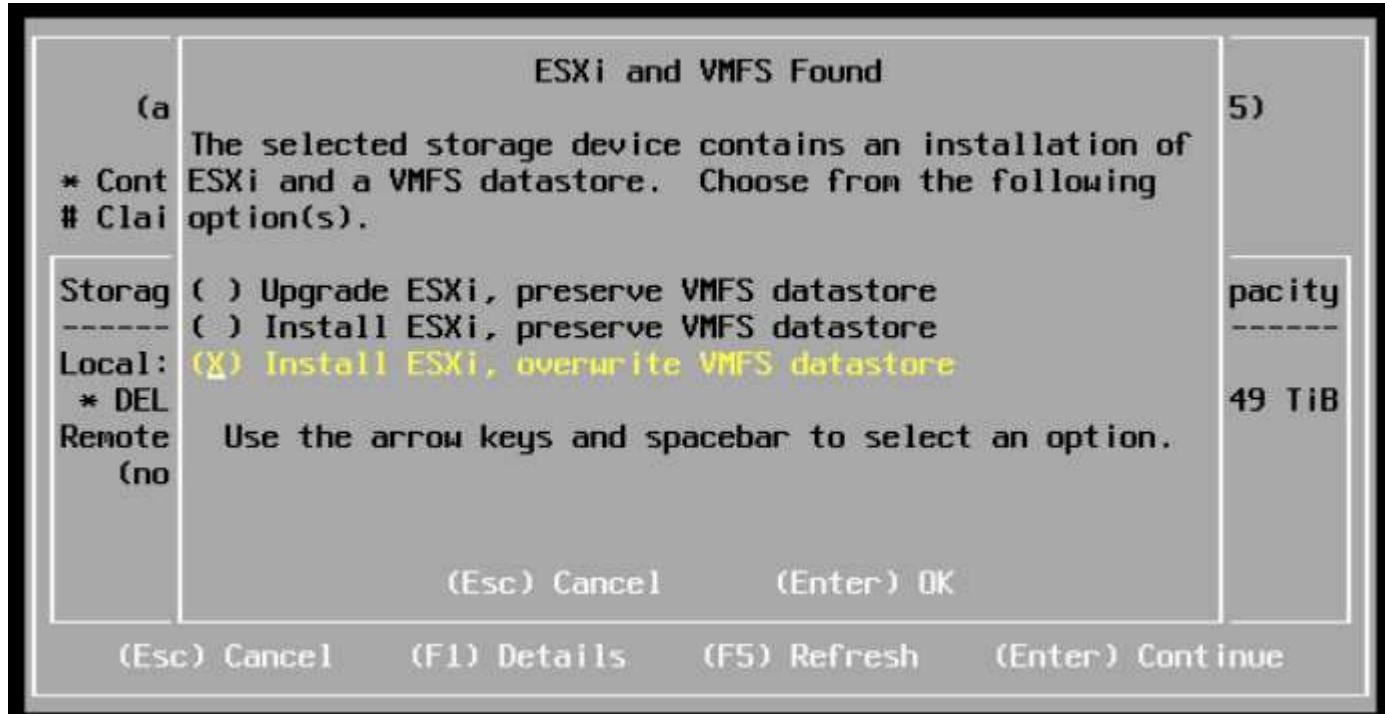
#### Note

You can install ESXi to a USB drive and then boot and run the system from that USB drive. This sample installation shows ESXi being installed on a local hard drive.

7. The installer scans the chosen drive to determine suitability for install



8. The Confirm Disk Selection window displays



9. Press [ENTER] to accept your selection and continue.

10. Please select a keyboard layout window displays.



11. Select your desired keyboard layout using the arrow keys and then press [ENTER]. The Enter a root password window displays.

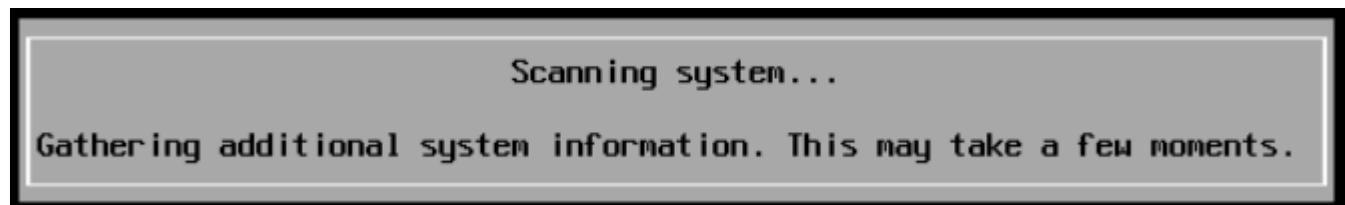


12. Enter a root password in the Root password field.

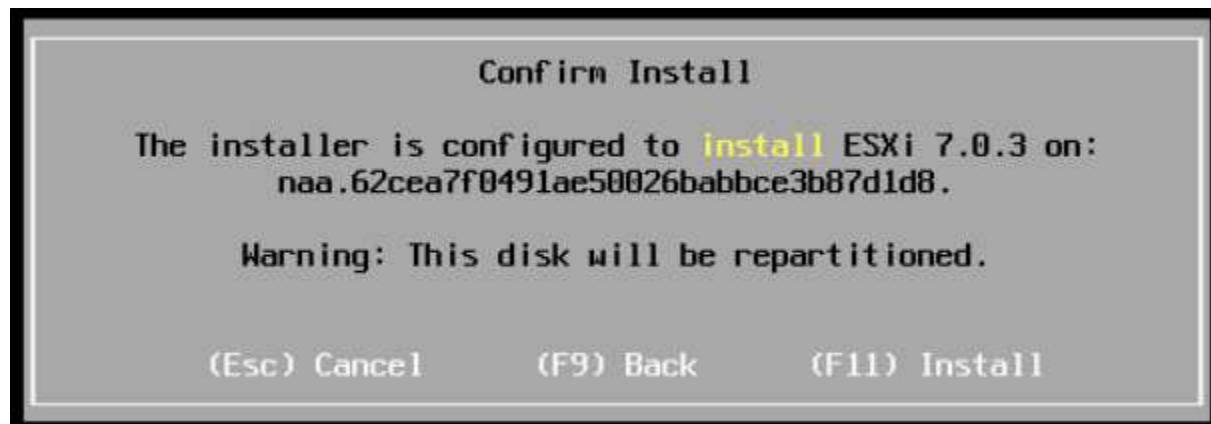
#### Important

To prevent unauthorized access, your selected root password should contain at least eight (8) characters and consist of a mix of lowercase and capital letters, digits, and special characters.

13. Confirm the password in the Confirm password field and then press [ENTER] to proceed. The installer rescans the system.



It then displays the Confirm Install window.



14. Press [F11] to proceed with the installation.

**Important**

The installer will repartition the selected disk. All data on the selected disk will be destroyed.

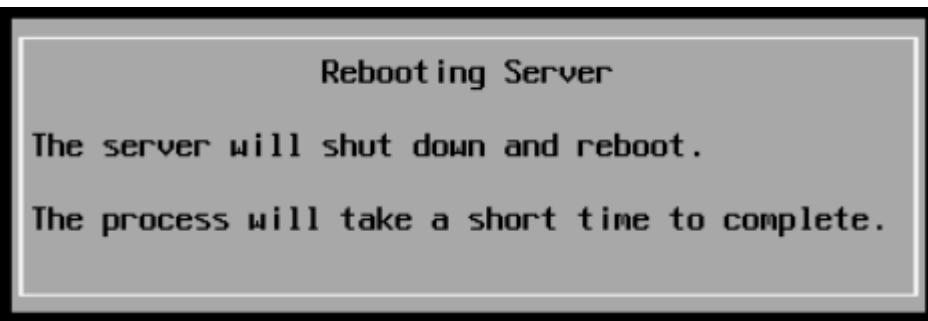
The ESXi installation proceeds.



15. The Complete Installation window displays when the installation process is completed.



16. Press [ENTER] to reboot the system. (Make sure your installation media has been ejected and your bios set to the boot disk.)



17. The installation is now complete.

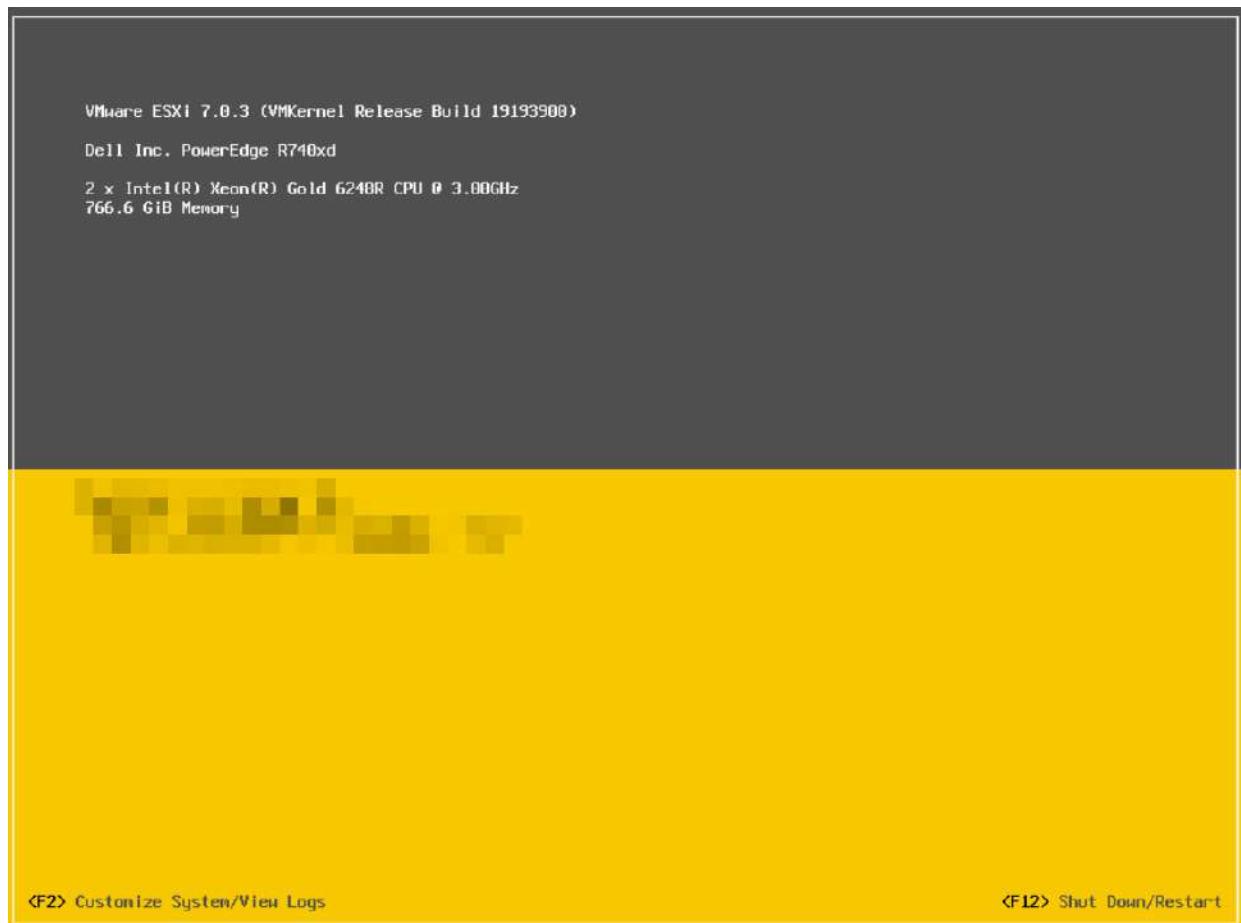
#### 4. Configure the ESXi Hosts as NTP Clients

##### Initial Host Configuration

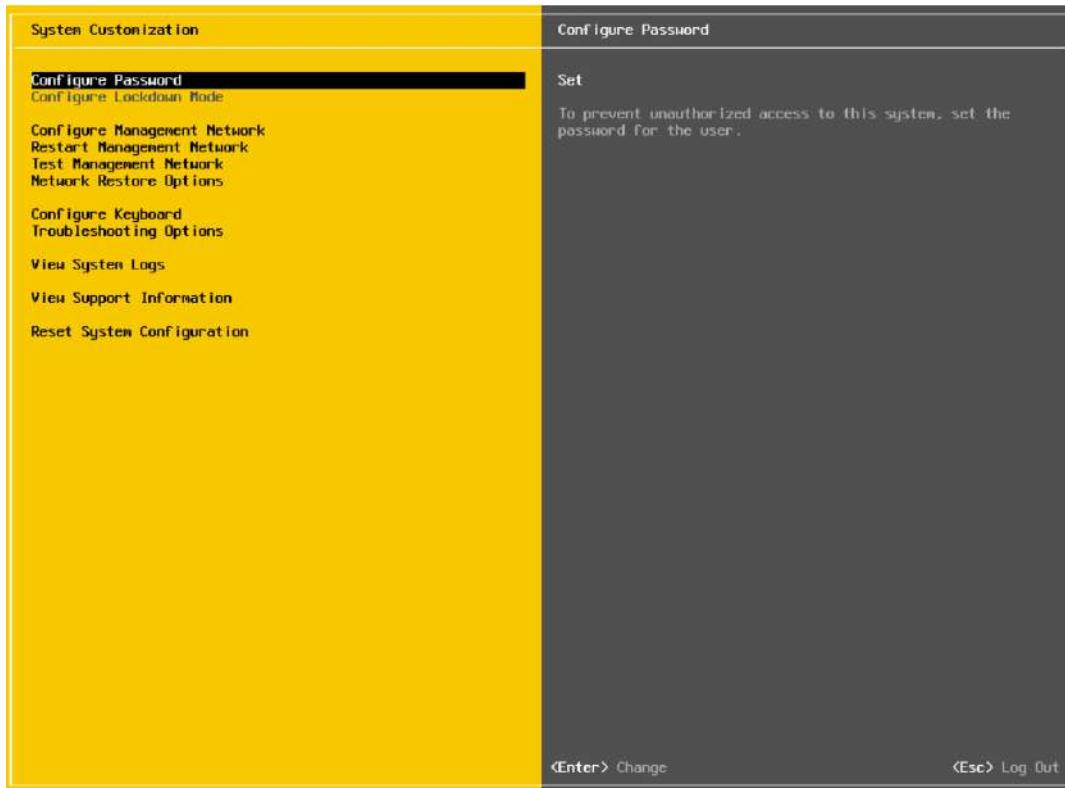
A countdown timer displays when you first boot ESXi. You can wait for the countdown to expire or press [ENTER] to proceed with booting. A series of notifications displays during boot, which can take several minutes to complete. The VMware ESXi screen displays when the boot completes.

Use the following procedure to configure the host:

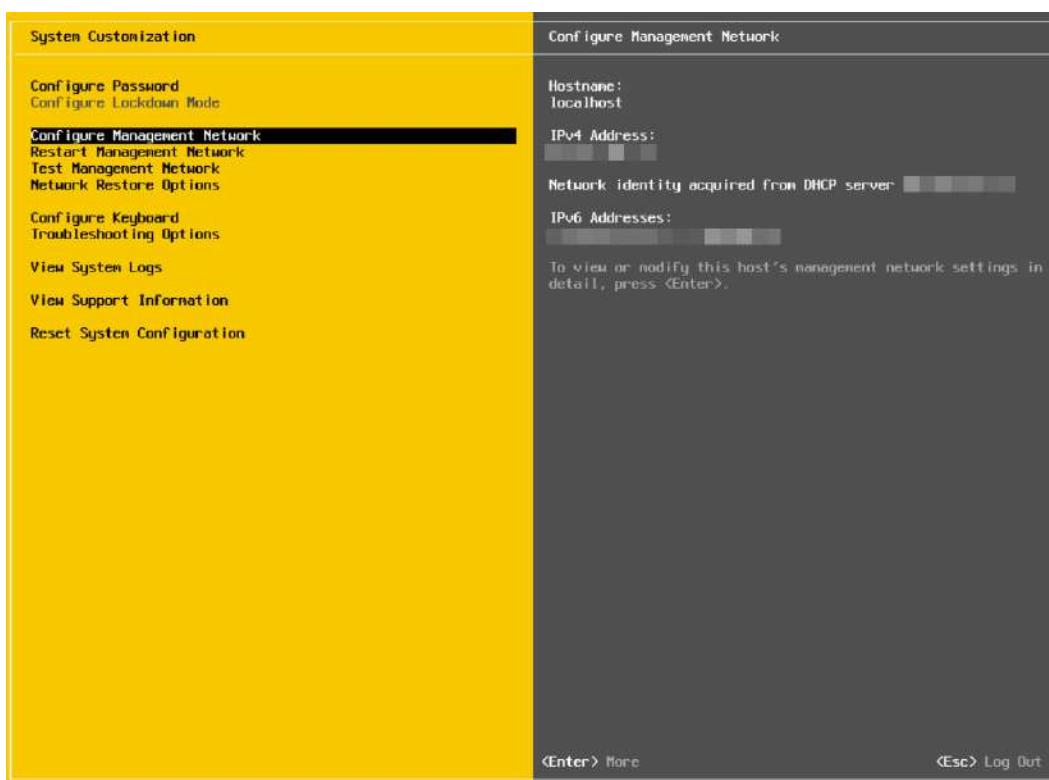
1. Press [F2]. The **Authentication Required** window displays.



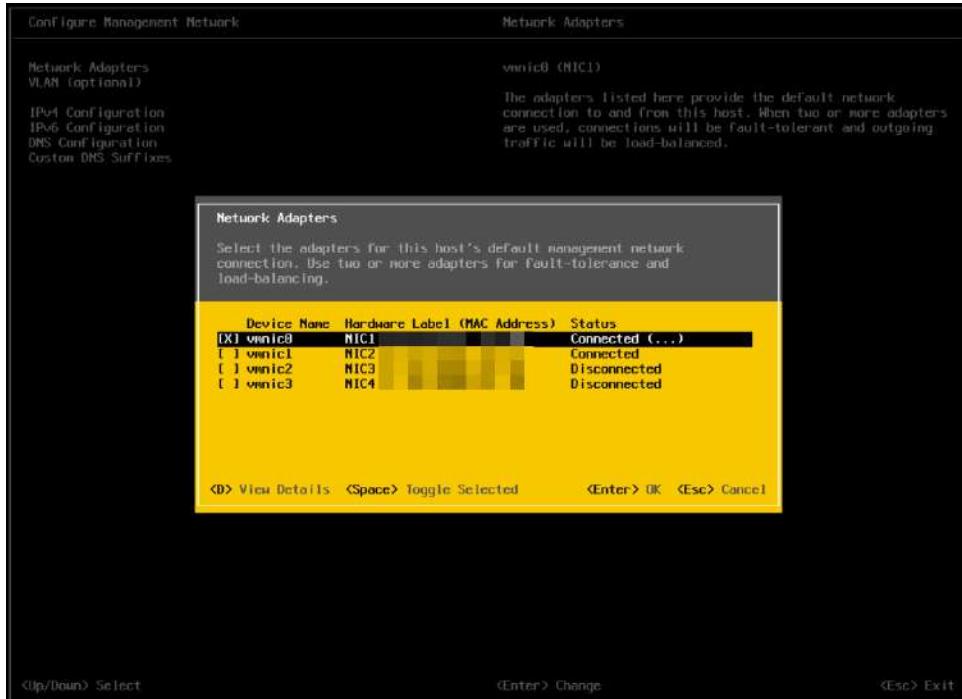
2. Enter the root account credentials you created during the installation process and press [ENTER]. The **System Customization** screen displays.



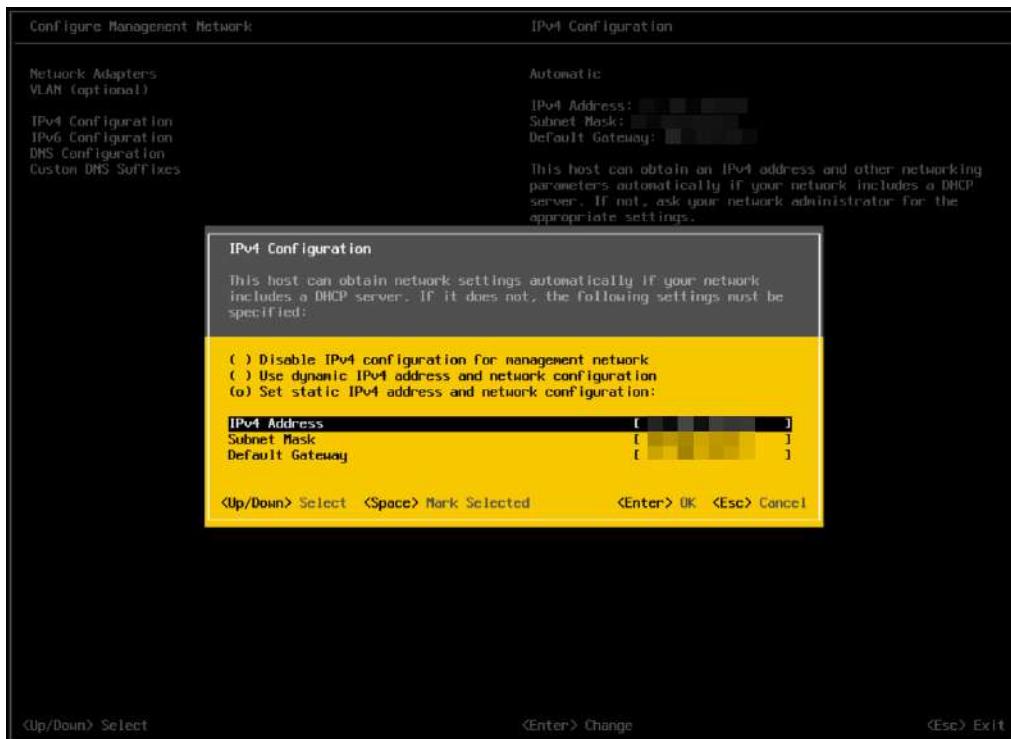
3. Scroll down to select **Configure Management Network** and then press **[ENTER]**.  
The **Configure Management Network** window appears.



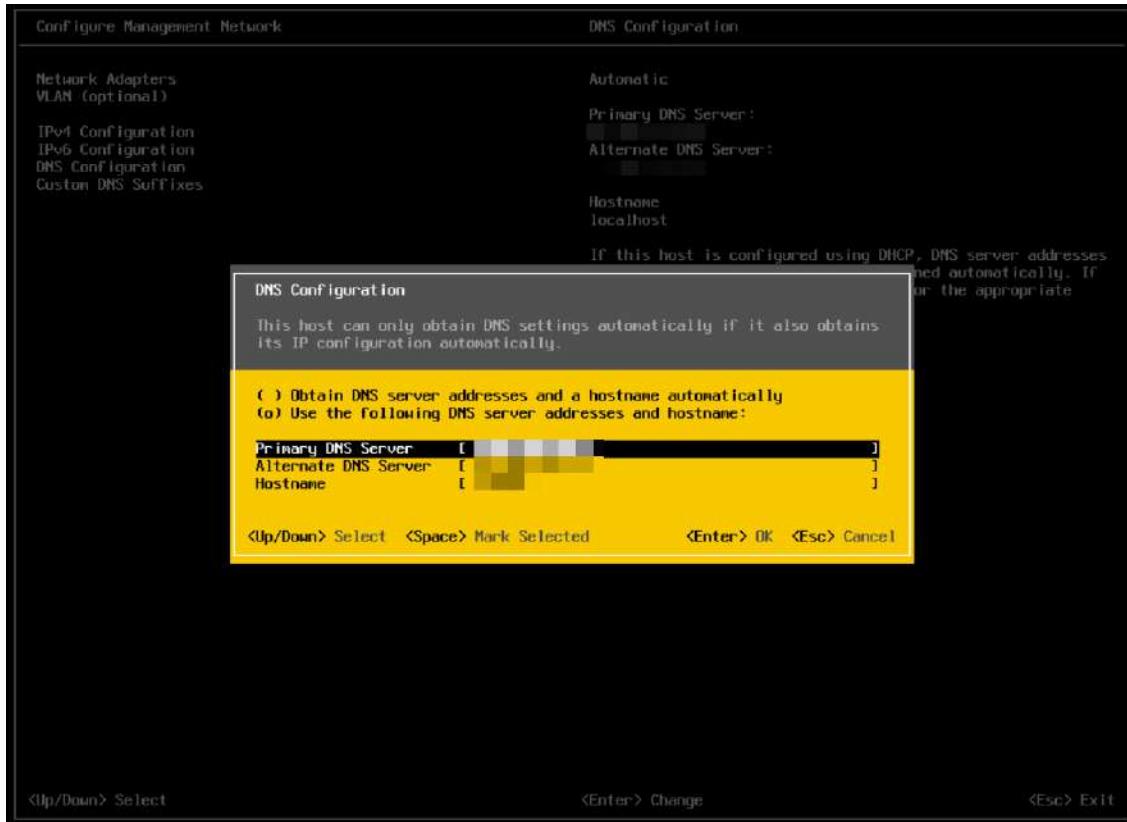
4. Select the Network Adapters window and press [ENTER]. Use the arrow keys to select the adapter to use as the default management network and press [ENTER]. More than one management network can be selected for redundancy.



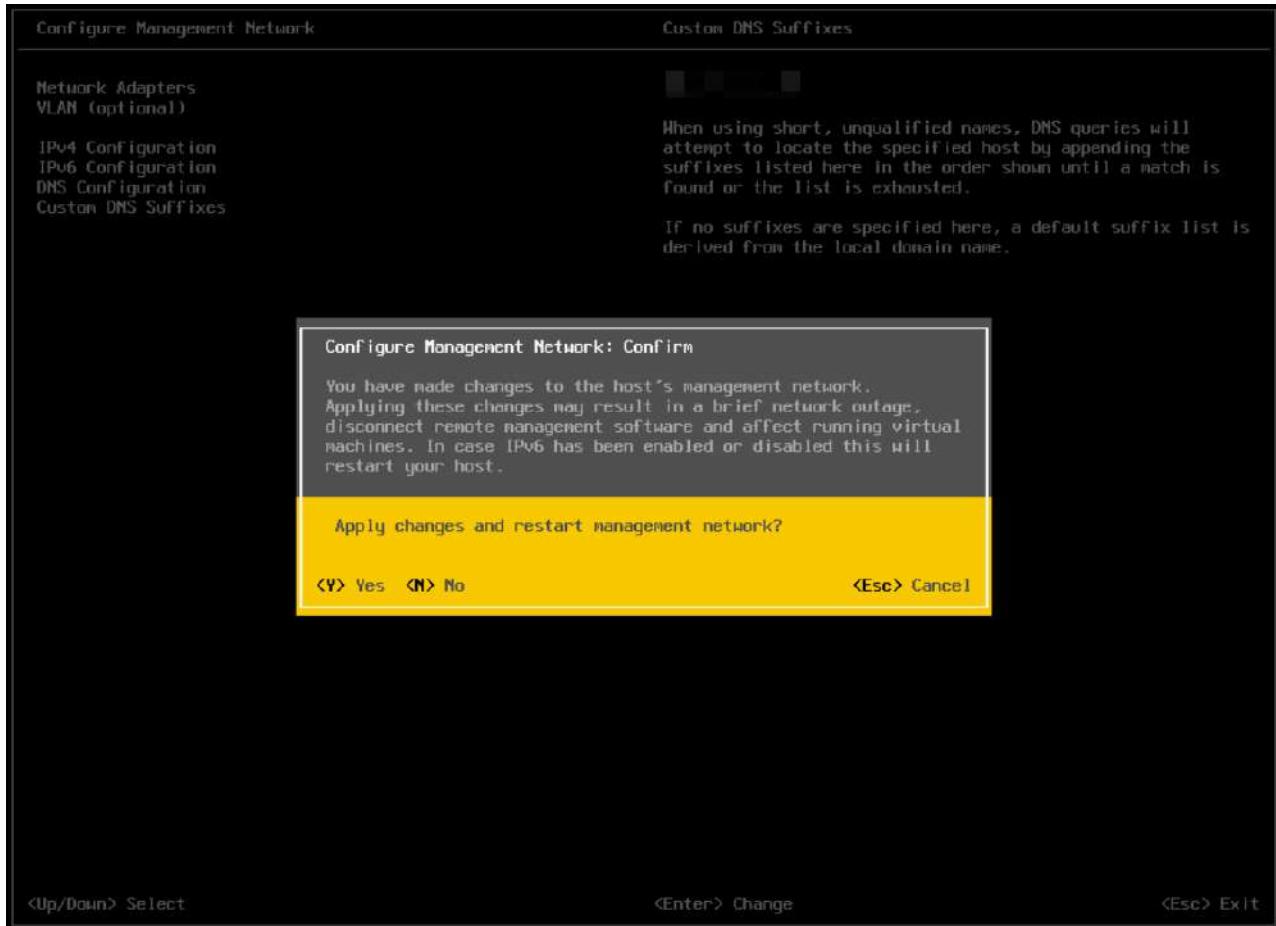
5. Exit the menu with [ESC] and select the IPv4 Configuration window



6. Use the arrow keys to select **Set static IPv4 address and network configuration** and then enter the IPv4 address, subnet mask, and default gateway in the respective fields.
7. Press **[ENTER]** when finished to apply the new management network settings.
8. Navigate to and select the **DNS Configuration window**. Add the primary and (if available) secondary DNS server address(es) in the respective fields. Set the host name for this ESXi host in the Hostname field.



9. Press **[ENTER]** to apply the new DNS settings and return to the Configure Management Network menu..
10. Press **[ESC]** to exit the Configure Management Network menu. The Confirm Management Network popup window displays. Press **[Y]** to confirm your selection.

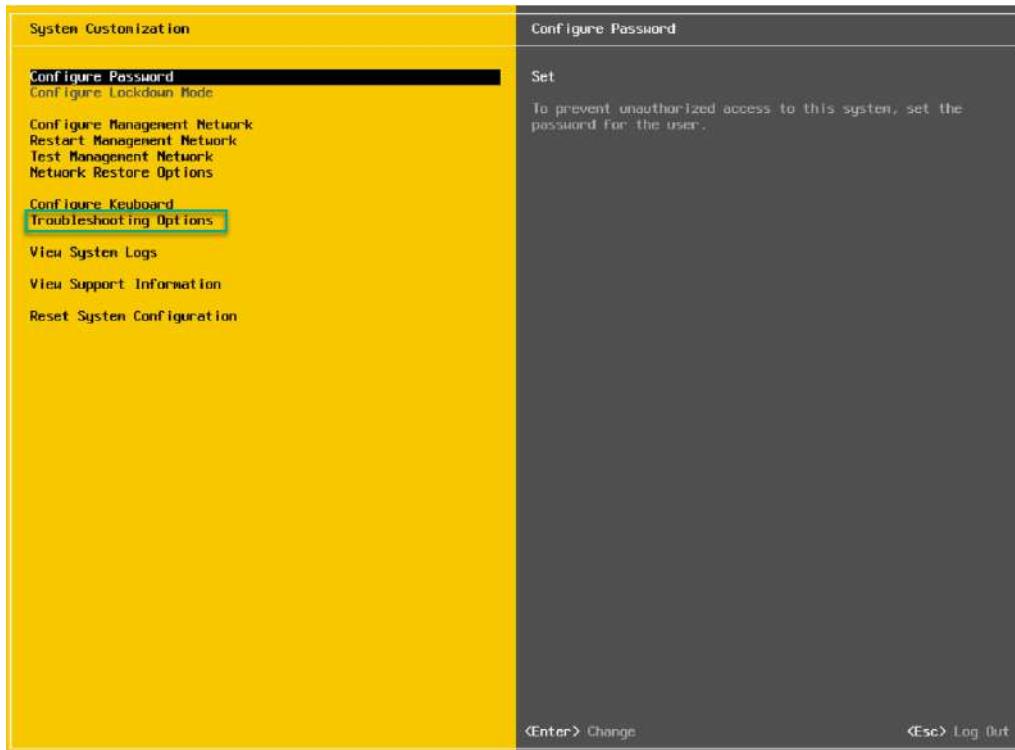


11. Select **Test Management Network** on the main ESXi screen to open the **Test Management Network** window.

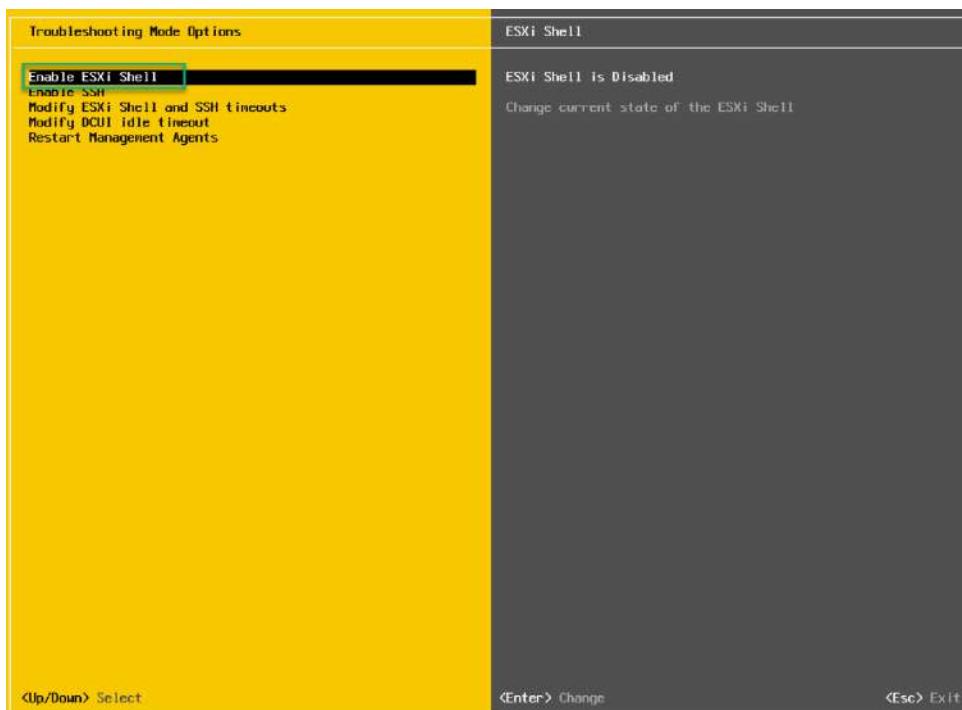
12. Perform the following tests:

- Ping the default gateway.
- Ping the DNS server.
- Resolve a known address.

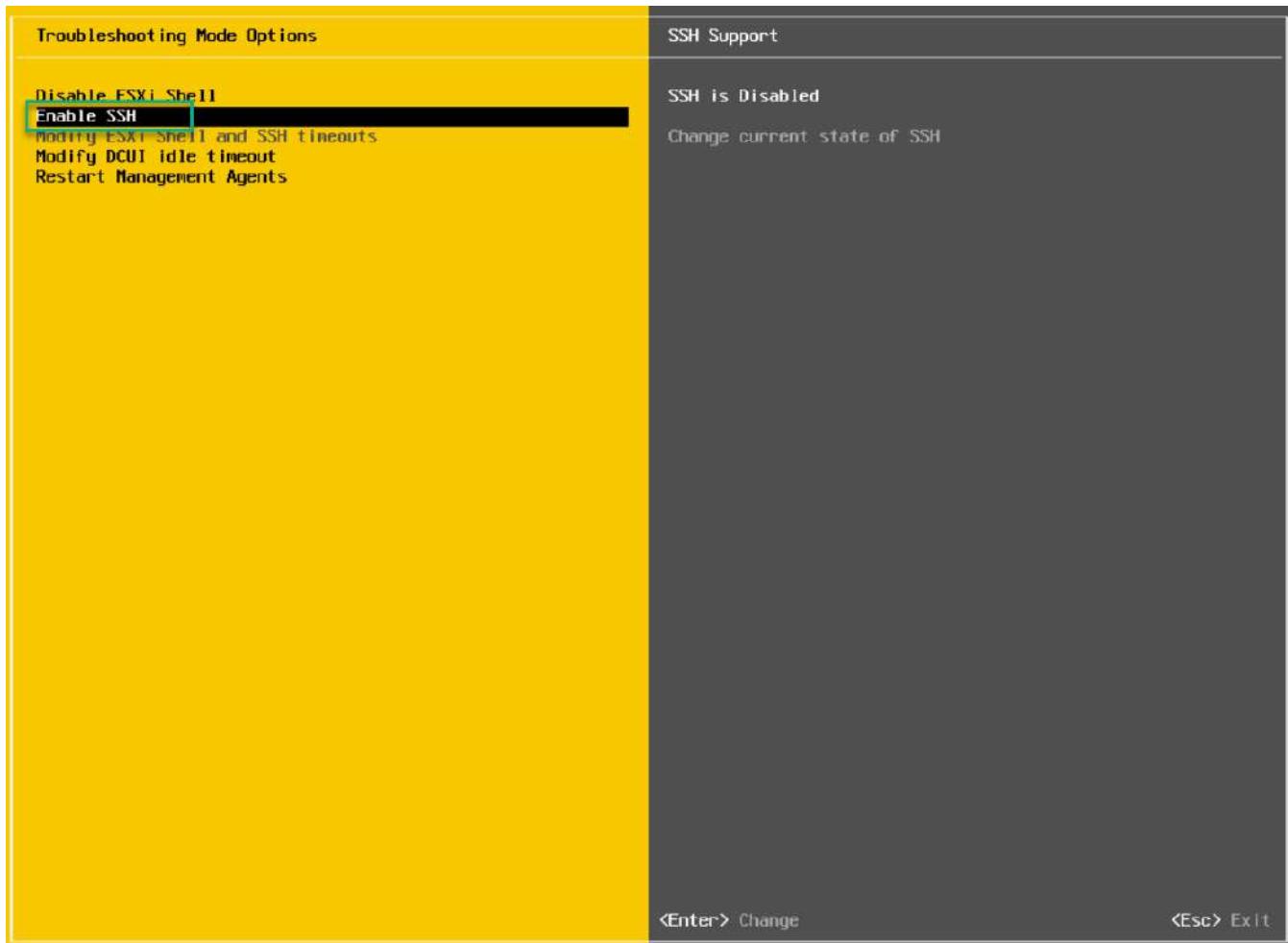
13. Return to the main ESXi screen when you have completed testing, and then select **Troubleshooting Options**. The Troubleshooting Mode Options window displays.



14. To install the NVIDIA VIB in a later step, you will need to enable the ESXi shell. This can be accomplished by selecting **Enable ESXi Shell**.
15. The window on the right displays the status: **ESXi Shell is Disabled**. Press [ENTER] to toggle **Enable ESXi Shell** on.

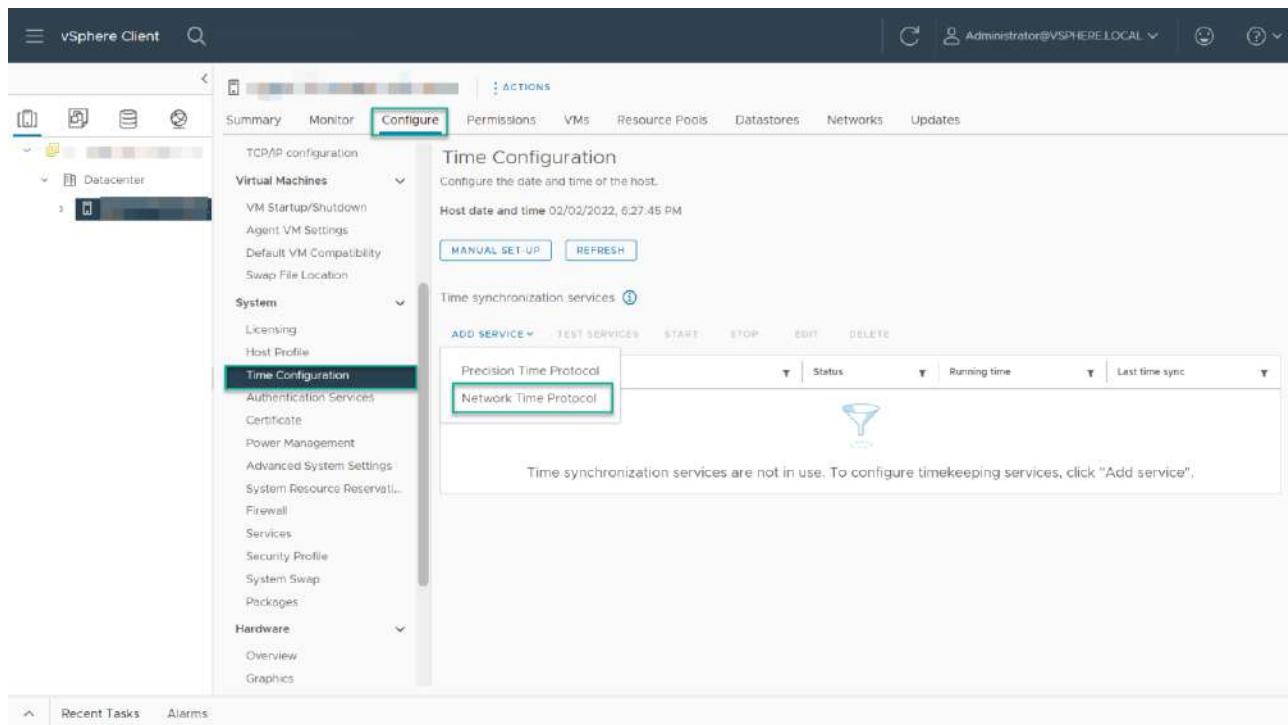


16. The window on the right displays the status: **SSH is Disabled**. Enable SSH by selecting **Enable SSH** and press **[ENTER]** to toggle this option on.



### Setting the NTP Service on a Host

1. Click a host object in the menu on the left, click **Configure > System > Time Configuration > Network Time Protocol > Edit**.



2. Check the **Enable** box and enter a valid time server and click **OK**.

## Network Time Protocol

X

Use Network Time Protocol (NTP) to synchronize the system time.

Enable monitoring events ⓘ

### NTP Servers

pool.ntp.org

If you enter multiple server names and IP addresses, use commas to separate them.

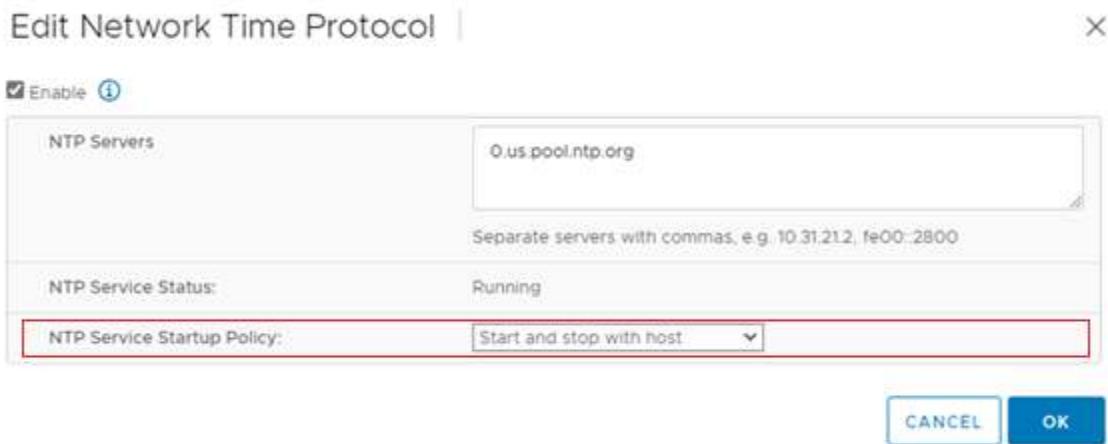
CANCEL

OK

## Note

Use a Public NTP Server, a recommended choice is pool.ntp.org

3. Set the **NTP Service Startup Policy** to **Start and Stop with host** and click okay.

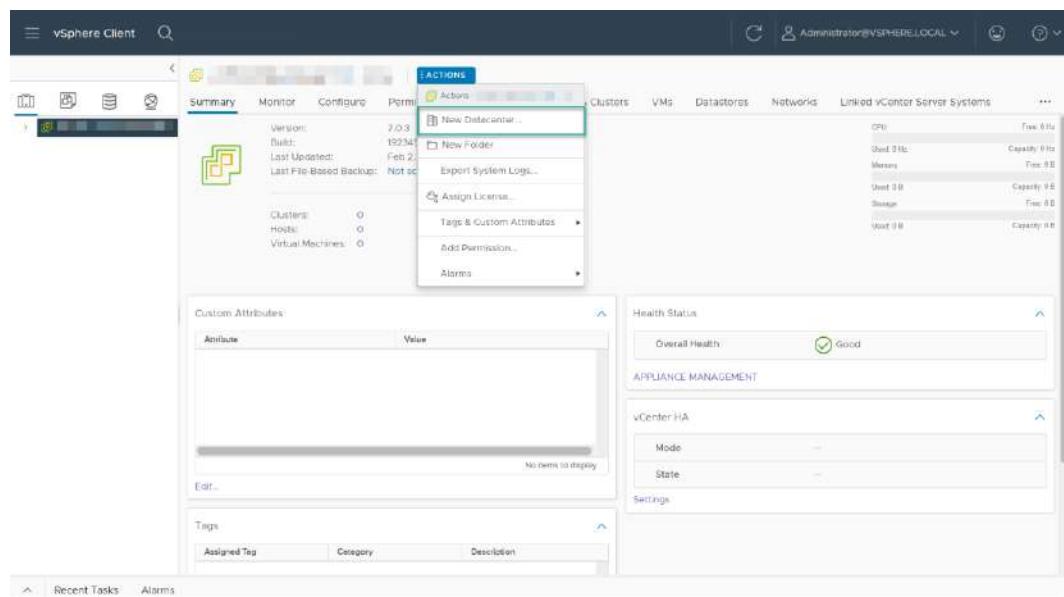


## 5. Create a Host and Cluster Folder

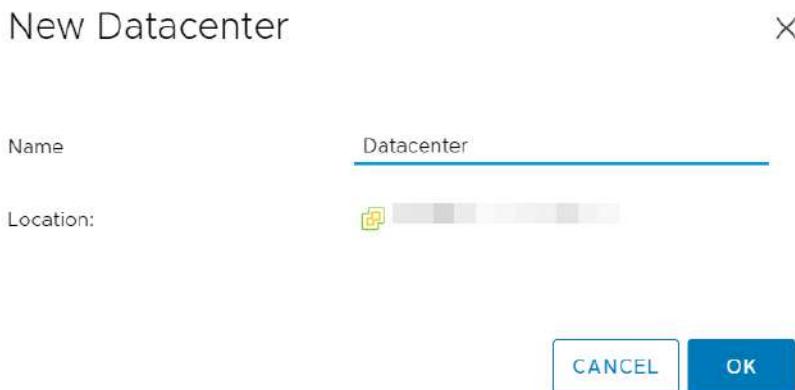
### Adding a Host

Use the following procedure to add a host in vCenter.

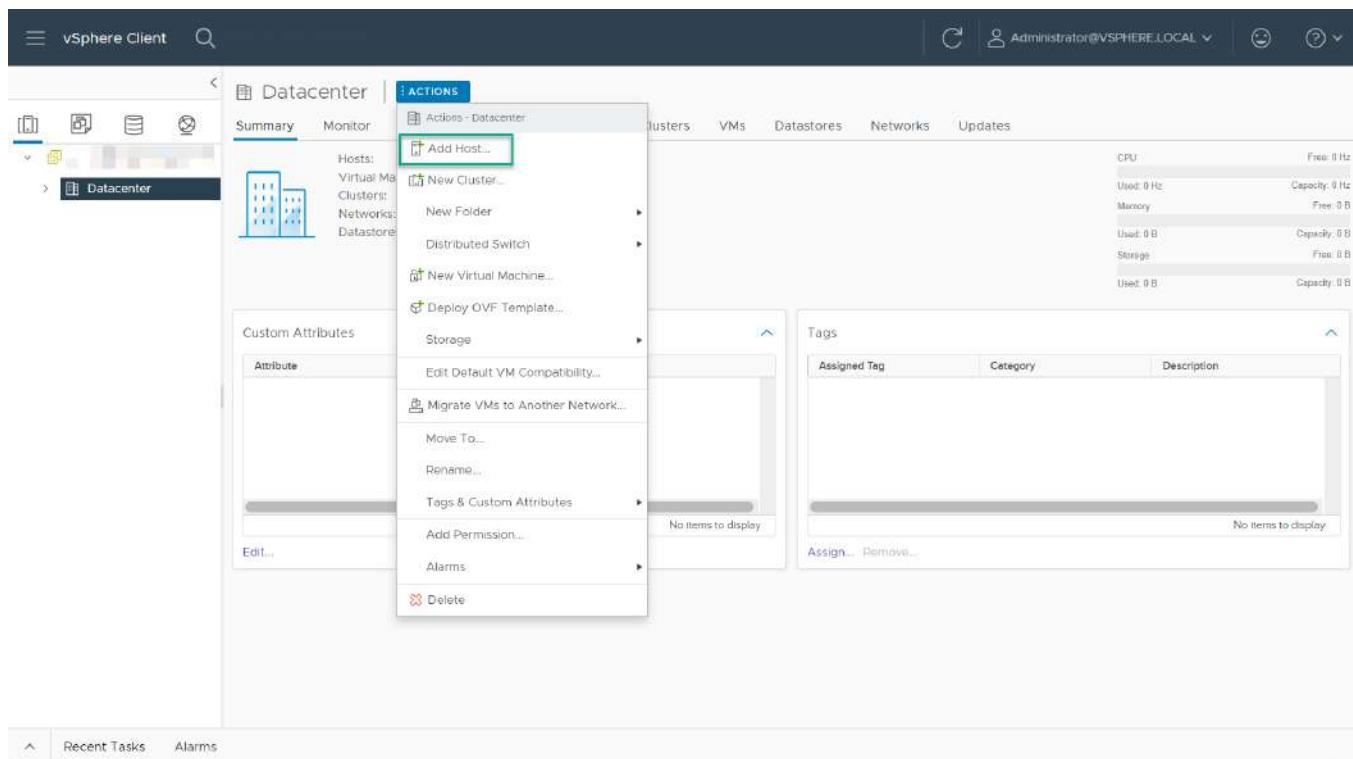
1. Select the **Home** icon (house) on the VMware vSphere Web Client page.
2. Select **Hosts and Clusters**.
3. From the **Actions** drop-down list, select **New Datacenter**.



4. Enter a name for the datacenter in the Datacenter Name field and click **Ok**.



5. The new datacenter is visible in the left panel of the vSphere Web Client. Click the **actions** drop-down and select **Add a Host**.



6. Enter the hostname or IP address of the vSphere host and click **Next**.

Add Host

1 Name and location

2 Connection settings

3 Host summary

4 Assign license

5 Lockdown mode

6 VM location

7 Ready to complete

Name and location

Enter the name or IP address of the host to add to vCenter Server.

Host name or IP address:

Location: Datacenter

CANCEL NEXT

7. Enter the administrator account credentials in the **Username** and **Password** fields and click **Next**.

Add Host

1 Name and location

2 Connection settings

3 Host summary

4 Assign license

5 Lockdown mode

6 VM location

7 Ready to complete

Connection settings

Enter the host connection details

User name:

Password:

CANCEL BACK NEXT

8. Click **Yes** to replace the host certificate.

## Security Alert

X

The certificate store of vCenter Server cannot verify the certificate.

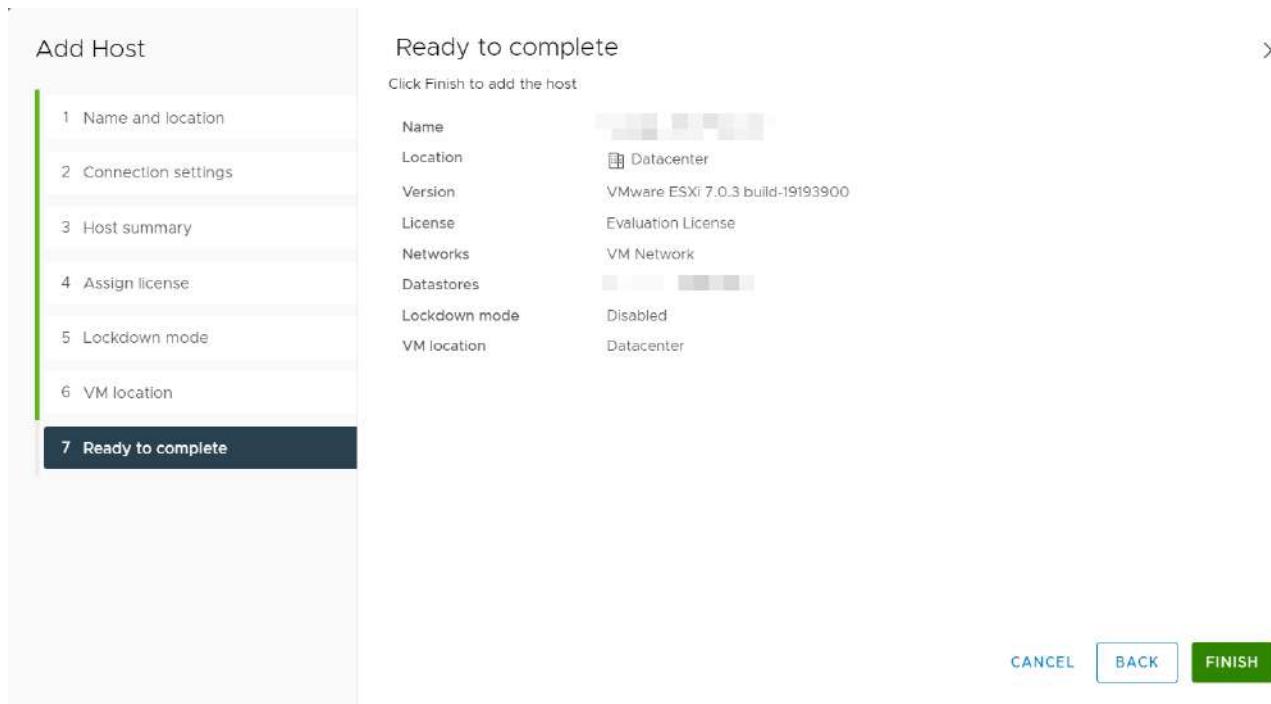
The SHA1 thumbprint of the certificate is:



Click Yes to replace the host's certificate with a new certificate signed by the VMware Certificate Server and proceed with the workflow.

Click No to cancel connecting to the host.

9. The Host summary dialog displays. Review the settings and click **Next** to proceed.
10. The Assign license dialog displays. Confirm the license selection and click **Next**.
11. The Lockdown mode dialog displays. Accept the default setting (Disabled) and click **Next**.
12. The VM location dialog displays. Select a cluster or accept the default option and click **Next** to continue.
13. The Ready to Complete dialog displays. Click **Finish** to complete adding the new host.



14. The new host is now visible in the left panel when you click the datacenter name.

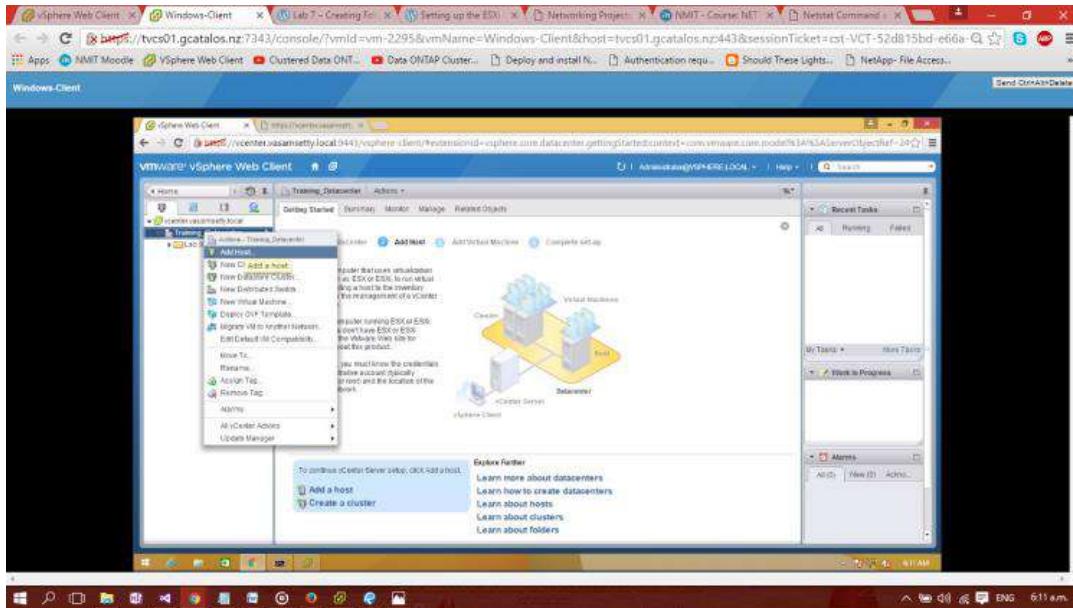
## 6. Create Virtual Machine and Template Folders.

### Create a Host Folder Object

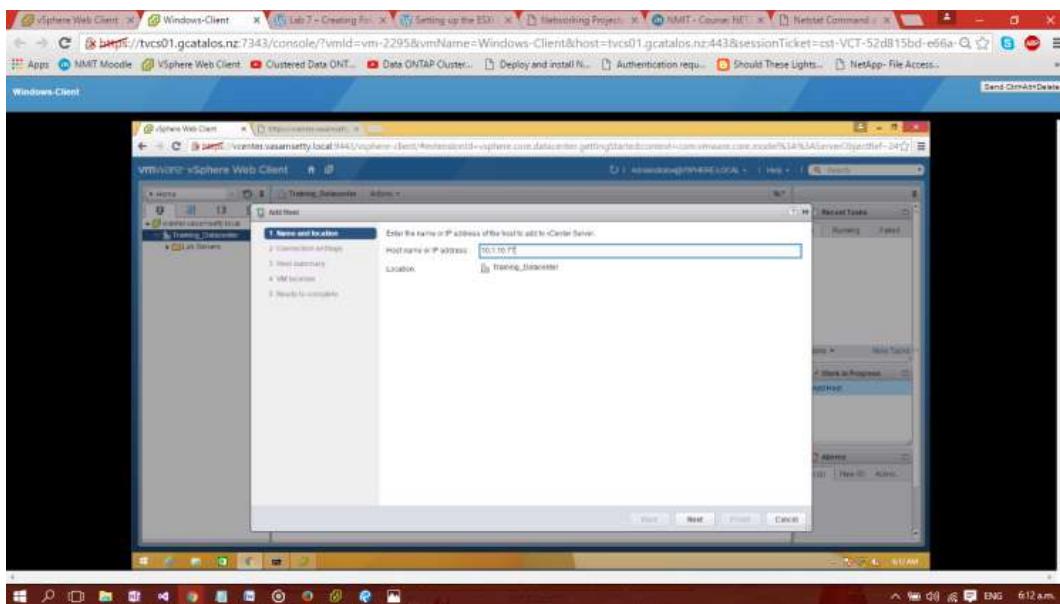
A folder can contain other folders, or a group of objects of the same type. For example, a single folder can contain virtual machines and another folder containing virtual machines, but it cannot contain hosts and a folder containing virtual machines.

You can create these types of folders: Host and Cluster folders, Network folders, Storage folders, and VM and Template folders.

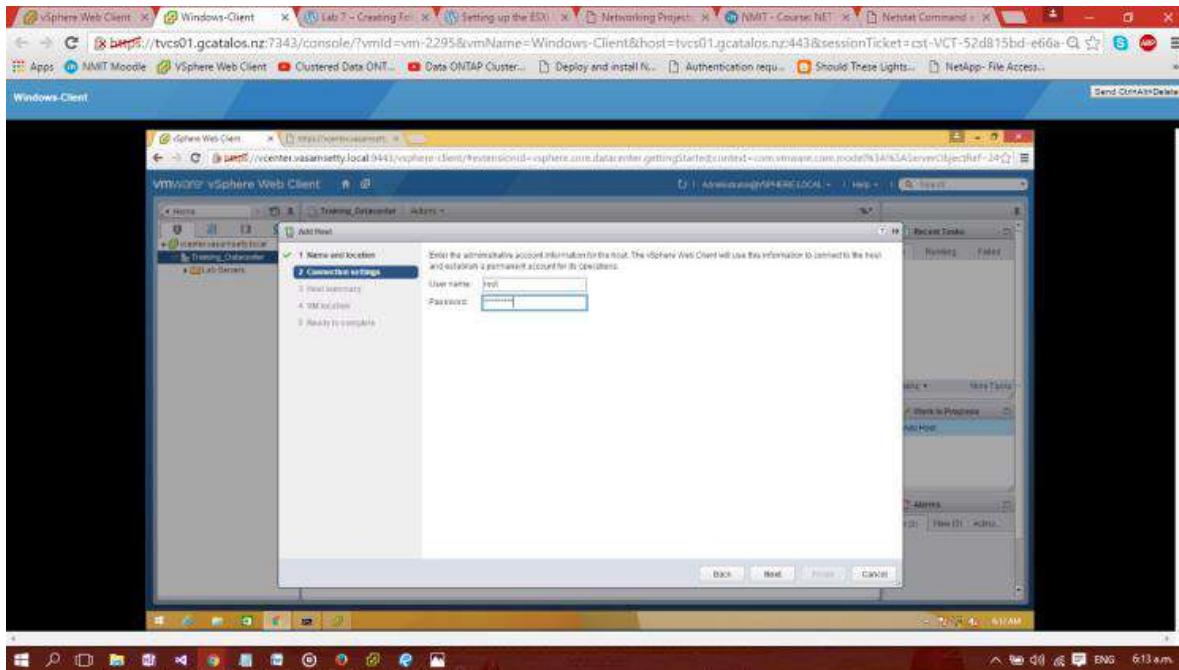
Lets see how to create a Host folder. Before creating a host folder make sure you have a valid host connected to your data center. For this Login into Vsphere web client url from your client device. Right Click on the Datastore that you created earlier and click on Add Host.



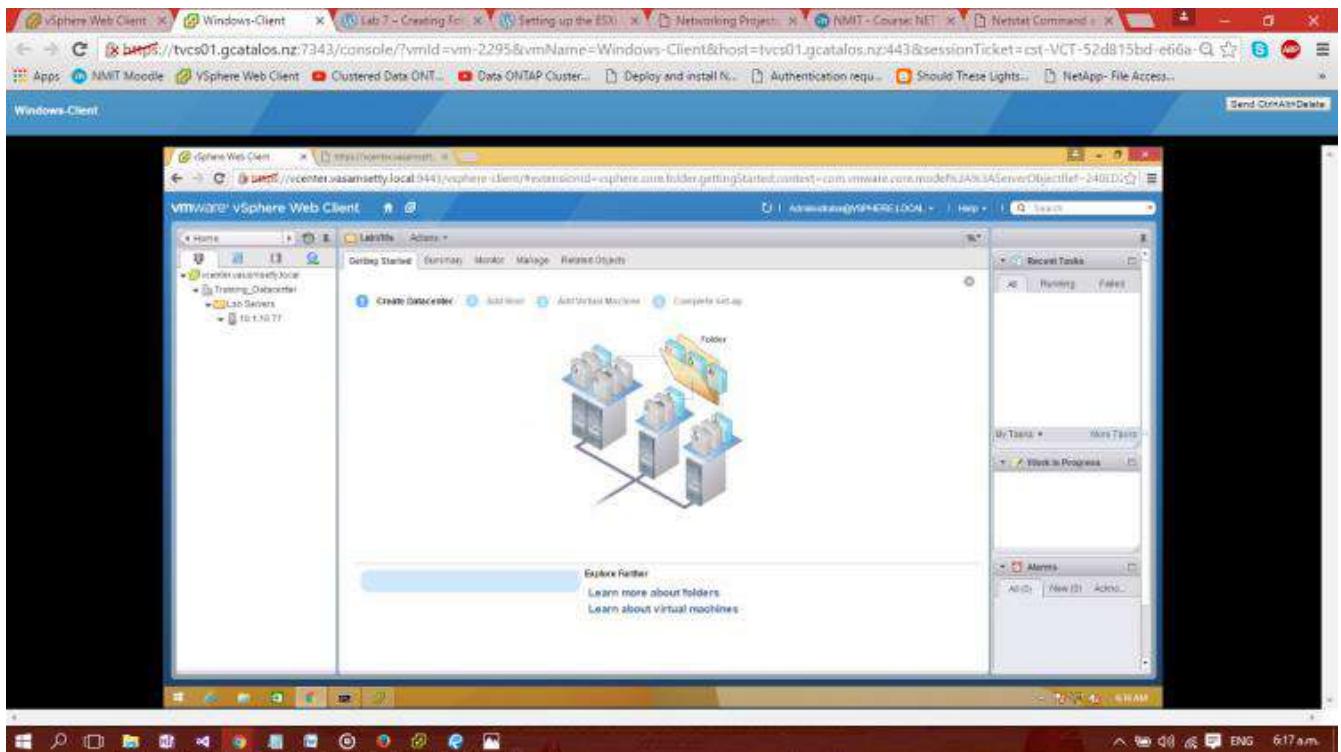
Enter your host IP address



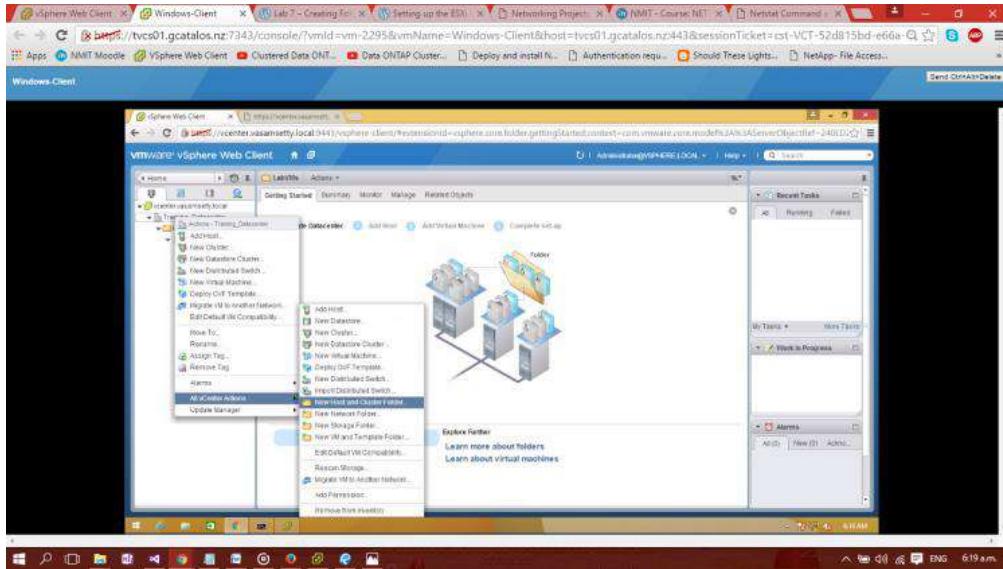
its ask for authentication provide the username and password. Give root as username and enter your master password.



once the authentication is approved you are followed with next steps host summary and licensing and vm location give defaults and click on finish. Once the host is ready you can see it on your datacenter dropdown as below.



Yup! host is ready now we gonna create a folder and add the host to that folder. Its so simple  
Right Click on **DataCenter**> and then **All Vcenter Actions>Create a Host and Clusters folder.**



give a name to the folder once the folder is been created drag the host into the folder and click yes to confirm.Thus a host folder is created. Now move to task-2

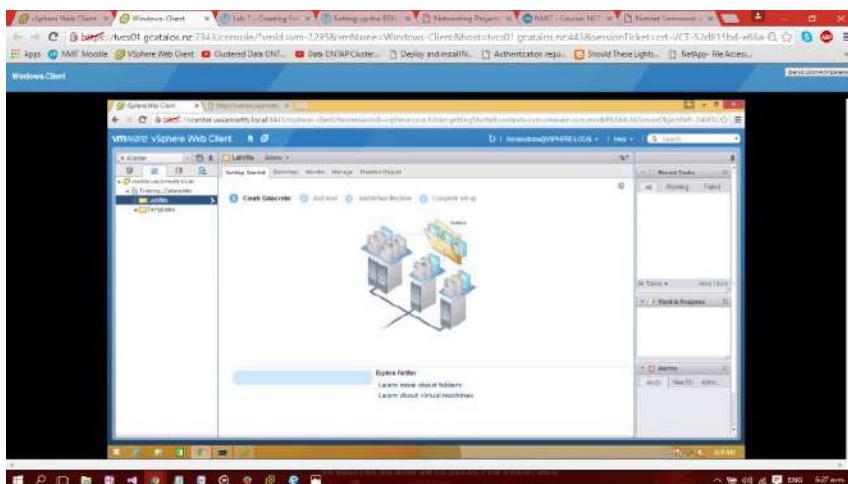
### Task-2 : Creating two virtual machine folders

this task is also three simple step process. just goto **Home>vCenter>VMs and Templates**

Right Click on **DataCenter**> and then **All Vcenter Actions>New VM and Templates Folder**

type one folder name as **Lab VM's** and other one as **Templates**

if you had done the steps correctly you can see as the below screen.



## 7. Navigate vSphere Client

### Installing vSphere Client

After installing the vSphere Hypervisor onto your server, you can then install the vSphere Client onto your Microsoft Windows® machine. Installing the vSphere Client, includes:

- Downloading the vSphere Client from VMware
- Assigning a License to VMware vSphere Hypervisor
- Assigning the network time servers
- Adding additional virtual network

To install the vSphere Client:

1. Open your web browser. Enter the IP address of the ESXi host which you configured in the procedure, Configuring vSphere Hypervisor. (<http://<ESXi host ip address>>) and press Enter. This accesses the web page to download the vSphere Client to your Windows machine. For example,

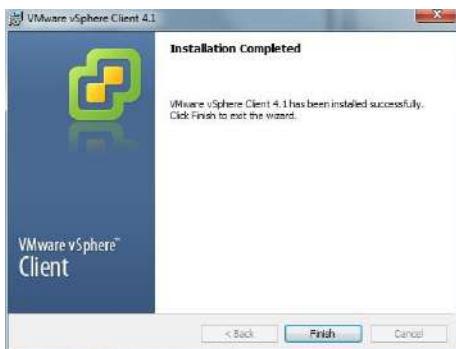
<http://172.30.58.164>

A warning displays followed by a prompt allowing you to accept or reject the certificate.

2. Press Enter to accept the certificate.
3. Click **Download vSphere Client**.
4. Navigate to the location on your PC where you downloaded the vSphere Client.
5. Double-click the file to begin the installation. The file proceeds to extract the application files and continues the installation process. The following screen displays.



6. Click **Next**. Select I agree to the terms in the license agreement and click **Next**. Continue the installation by following all remaining instructions for installing the vSphere Client. When the installation is complete, the following screen displays.



7. Click **Finish** to complete the installation. The VMware vSphere Client icon appears on your PC desktop.



8. Double-click the VMware vSphere Client icon. The following screen displays.



9. In the **IP address / Name** text box, enter the IP address or the domain name of the ESXi host. For example:

IP address / Name: **172.30.58.164**

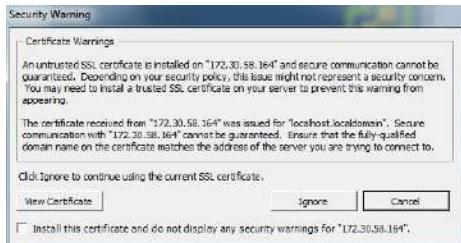
10. In the **User name** text box, enter the user name assigned to you by the system administrator of the ESXi host. For example:

User name: **root**

11. In the **Password** text box, enter the password assigned to you by the system administrator of the ESXi host. For example:

Password: **jre453i**

12. Click **Login**. The following Security Warning displays:



13. Place a check mark in the box that indicates: Install this certificate and do not display any security warnings for <ip\_address>. The IP address is the address of the ESXi host.

14. Press **Ignore**. The VMware Evaluation Notice alert displays.

15. Click "Assign license to the ESXi host."

**Note:**

vSphere 4 Hypervisor is licensed for 2 physical CPUs (free, never expires).

Getting the vSphere Hypervisor License

To get the VMware vSphere Hypervisor License:

1. Enter the following URL:

<https://www.vmware.com/account/login.do>

2. Register for a VMware account by clicking Register. Or if already registered, enter your email address or VMware customer number, and password, and click Sign In.

VMware sends the following message to the email address you specified during registration:

Thank you for creating a VMware account. To complete the registration process, please click the button below.

3. Open your email message from VMware and click the Activate Now button.

The VMware 's Enter Your Password screen displays.

4. In the Password text box, enter the password you specified when registering with VMware and click Continue.

The "Account Activated" screen displays with the following message:

Success! Your account has been activated.

5. Copy and paste the following link into your browser:

<https://my.vmware.com/web/vmware/evalcenter?p=free-esxi5&lp=default&lp=1&ie=UTF-8&q=vmware%20vsphere%20hypervisor%20esxi%204.1%20license>

6. In the box, **On how many physical servers do you plan to install VMware vSphere Hypervisor?**, enter the number of servers on which you are installing the VMware vSphere Hypervisor.

Valid values are 1 - 999.

7. Place a check mark in the box, **I agree to the terms and conditions outlined in the VMware vSphere Hypervisor End User License Agreement.** and click **Register**.

VMware sends you an email message for accessing your VMware ESXi License.

8. Open your email message from VMware and click **Access Now**.

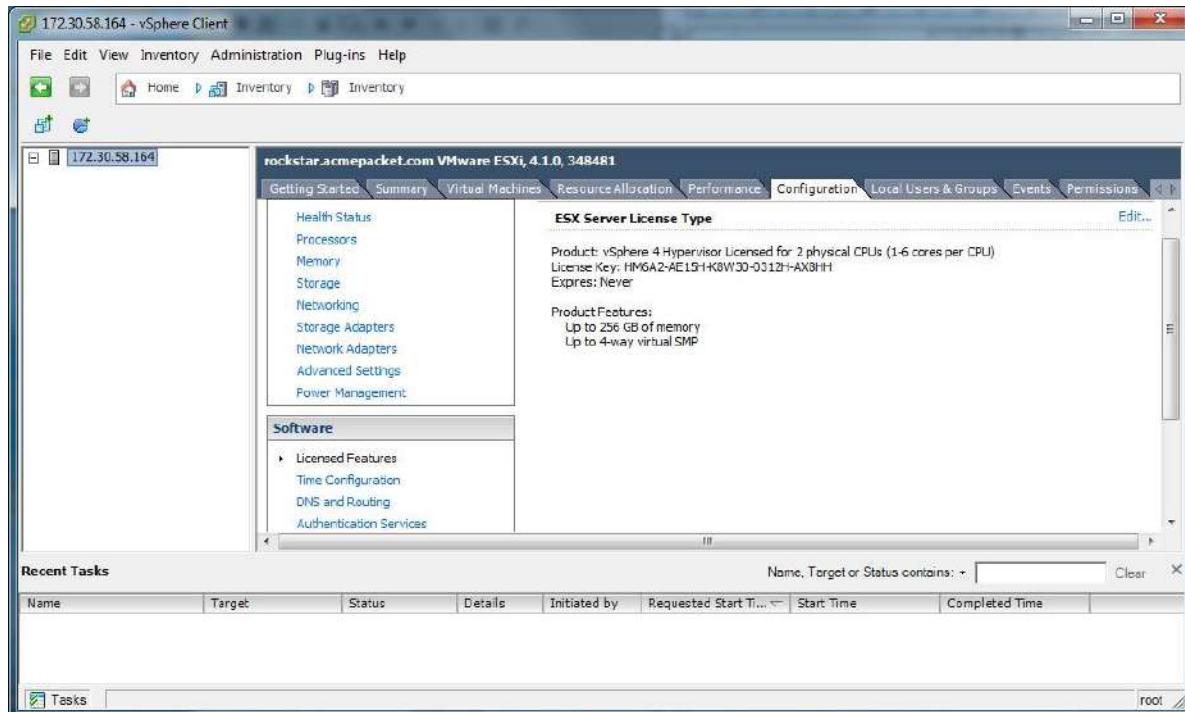
A VMware vSphere Hypervisor license string displays.

9. Copy the VMware vSphere Hypervisor license key string.

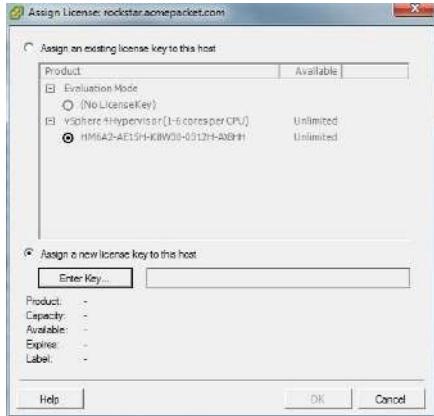
#### Applying the VMware vSphere Hypervisor License

To apply the VMware vSphere Hypervisor license:

1. Click **OK** to close the VMware Evaluation Notice window that displayed in Step 14. The following window displays. The ESXi Host IP displays in the left column.



2. Click the **Configuration** tab.
3. In the left column, under the Software category, click **Licensed Features**.
4. In the upper right corner of the window, click **Edit**. The following window displays.



5. Click the radio button **Assign a new license to this host**. and click **Enter Key**. A pop-up displays allowing you to enter the license key string.
6. In the **New License Key** text box, paste the license key string you copied from Step 24. Or enter the license key string manually.

### Module 3: Users, Groups & Permissions

#### How to Join vCenter to an Active Directory Domain

To integrate a vCenter Server Appliance (VCSA) with Microsoft Active Directory as the identity source simplifies and improves the security of access management. By joining vCenter to an AD domain, VMware vSphere administrators can use the same identity source used to grant access to file servers and other resources on the network to grant access to vSphere objects. Read on to learn the steps for how to join vCenter to domain.

#### VMware vSphere Backup

Complete data protection for VMware vSphere VMs and instant recovery options. Secure backup targets onsite, offsite and in the cloud. Anti-ransomware features.

#### How to Add vCenter to an Active Directory Domain

Active Directory is a common standard for the centralized authentication of users in many organizations. Active Directory can also be used to authenticate VMware ESXi and VMware vCenter users. Then we can assign the needed vSphere permissions for authenticated Active Directory domain users.

#### Requirements

There are some requirements for configuring vCenter AD integration:

- An Active Directory domain controller must be configured. The domain controller must be writable (and not just in read-only mode).
- The DNS suffix used for a fully qualified domain name (FQDN) of the vCenter Server must be correct.
- The DNS settings of VCSA to communicate with the domain controller must be correct.
- vCenter Server Appliance (VCSA) must resolve the DNS name of the Active Directory domain controller to an IP address.

**Note:** It is also possible to join a standalone ESXi host to the AD domain.

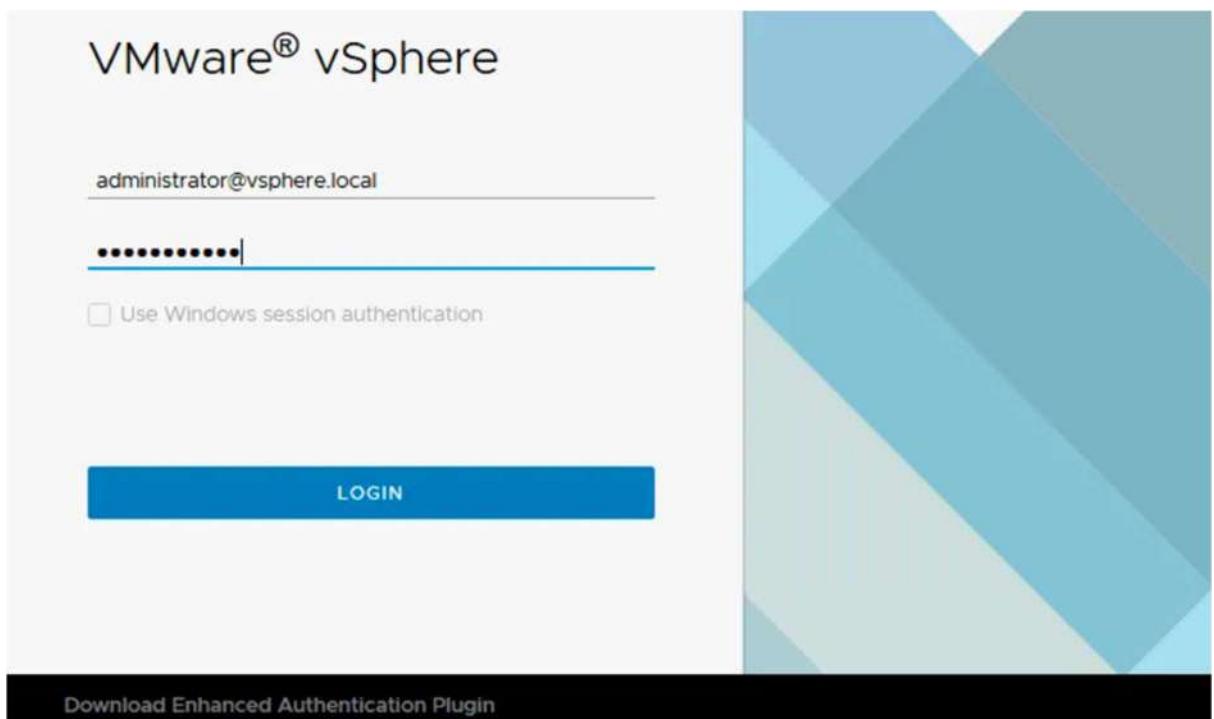
#### How to join vCenter to domain

We need to join our VCSA appliance to Active Directory as an object in order to enable **Active Directory (Integrated Windows Authentication)**. This option allows us to pass the logged-on user's Windows credentials as authentication to the vCenter Web Client. Note that in this tutorial, we are using vCenter Server Appliance 7.0 with an embedded platform service controller.

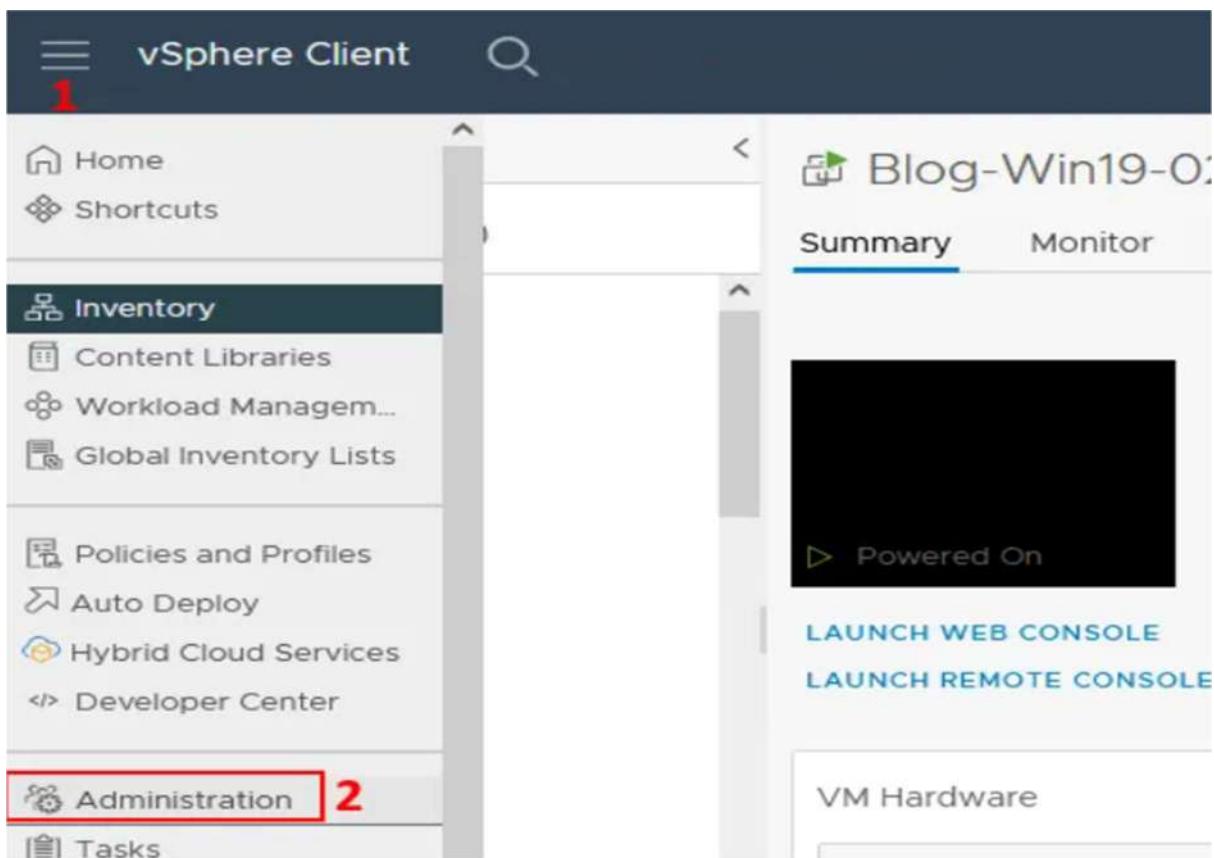
Complete the following steps to set up the AD connection:

1. Log in as the **SSO administrator** to vCenter by using a web browser and going to the VMware vSphere Client page. The default administrator name is *administrator@vsphere.local* (covered in a previous post on [vSphere SSO domain](#)), which is the admin user set up during VCSA installation. Also, keep in mind, this is not a Windows

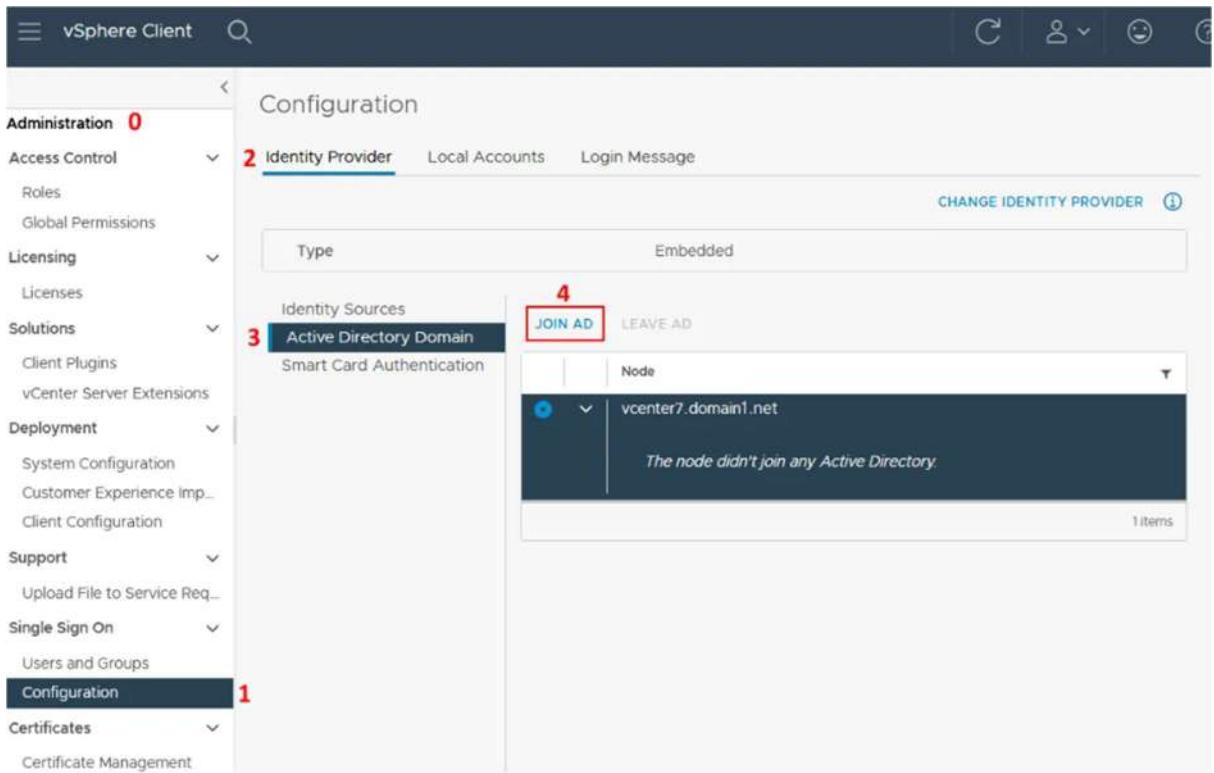
Active Directory domain user. However, you can use the *Use Windows session authentication* option when you integrate vCenter with Active Directory.



- Once you are logged in to the Web Client as the SSO administrator, click the menu icon in the top left corner. Click **Administration** in the menu that opens.



3. Click **Configuration** on the *Administration* page in the *Single Sign On* section. Select the **Identity Provider** tab, click **Active Directory Domain**, and click **JOIN AD** to join vCenter to domain.



4. This will bring up a dialog box to type in the **Domain**, **Organizational unit**, **Username**, and **Password**.

- Enter the Active Directory domain name, for example, *domain1.net*. Note that the name of your existing local SSO domain (*vsphere.local* in our case) and the Active Directory domain (*domain1.net* in our case) must be different. If you use the same AD domain name, you get an error and won't be able to join the domain and integrate vCenter with Active Directory.
- Setting an organization unit can be useful for those who are familiar with LDAP. If the *Organization unit* field is left empty, then a computer account in AD is created in the default location, which is a *Computers* container. You can always move a computer object to the needed organization unit on your Active Directory domain controller. An example of how to fill the Organization Unit field:

*OU=Unit1,DC=domain1,DC=net*

- Enter the username of the Active Directory domain administrator and the password. Our domain administrator is *administrator@domain.net*. However, you can create a dedicated user (for example, *vmwareadmin*) on the domain controller and add this user to the appropriate domain administrators group.

After completing the dialog box, click **Join** and you will be prompted to reboot your vCenter appliance.

## Join Active Directory Domain

X

Domain

Organization Unit (optional)

Username

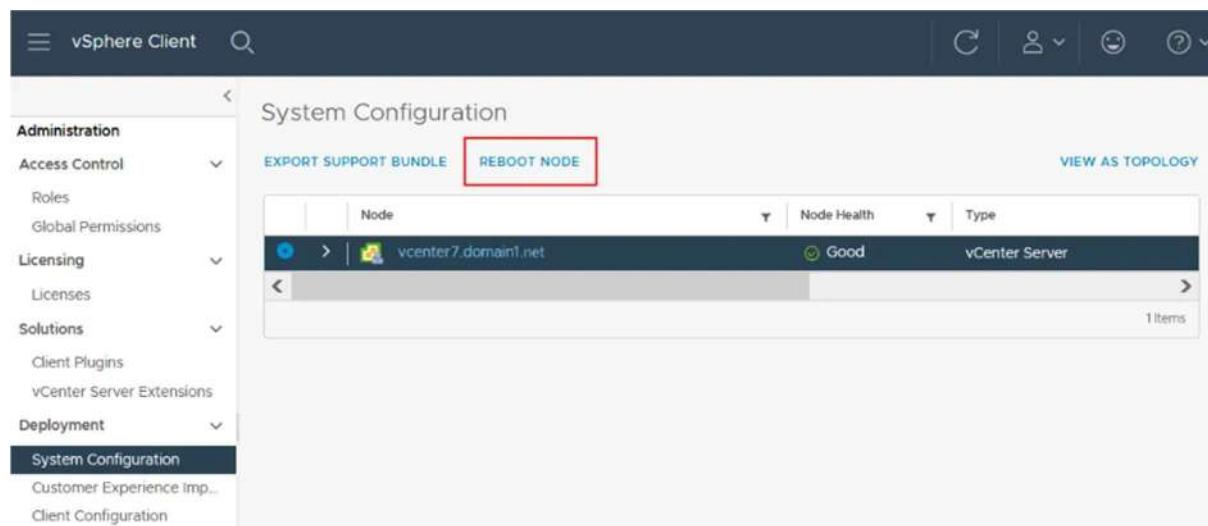
Password

(i) Reboot the node to apply changes.

CANCEL JOIN

5. To reboot vCenter server from the interface of VMware vSphere Client, go to **Administration > System Configuration**, select your vCenter node, and click **Reboot node**.

Alternatively, you can log in to VMware Host Client of the ESXi host on which the vCenter Server Appliance VM is running and reboot the VCSA VM. Another solution is to use the Direct Console user interface (DCUI) on the VCSA and use the Reboot option there.



6. After the vCenter reboot, you can go to **Administration > Single Sign On > Configuration > Identity Provider > Active Directory Domain** (as you did earlier) and ensure that connecting to domain controller is successful and your vCenter is a domain member now.

The screenshot shows the vSphere Client interface with the 'Configuration' screen open. The left sidebar has 'Administration' expanded, with 'Identity Provider' selected. The main pane shows the 'Identity Provider' tab selected. Under 'Identity Sources', 'Active Directory Domain' is highlighted. To the right, there's a tree view labeled 'JOIN AD' and 'LEAVE AD' with 'vcenter7.domain1.net' under 'Active Directory'. The status bar at the bottom right says '1 items'.

7. You can also ensure that your vCenter machine has joined the domain in Windows Server acting as a domain controller. For this purpose, open **Active Directory Users and Computers**, select your domain, and click **Computers**. You can see that our **vcenter7** machine is a member of our Active Directory domain in the screenshot below.

The screenshot shows the 'Active Directory Users and Computers' management console. The left pane shows the navigation tree with 'domain1.net' expanded, showing 'Saved Queries', 'Builtin', 'Company' (with 'unit1' and 'unit2' children), 'Computers' (which is selected and highlighted in grey), 'Domain Controllers', 'ForeignSecurityPrincipal', 'Managed Service Account', and 'Users'. The right pane lists 'Computers' with two entries: 'SERVER09' and 'vcenter7'. Both entries have a computer icon next to them, 'Type' listed as 'Computer', and 'Description' listed as 'vcenter7.domain1.net'.

### Adding the identity source

After the vCenter Server Appliance (VCSA) has been joined to the domain and rebooted, we are now ready to add our Active Directory identity source:

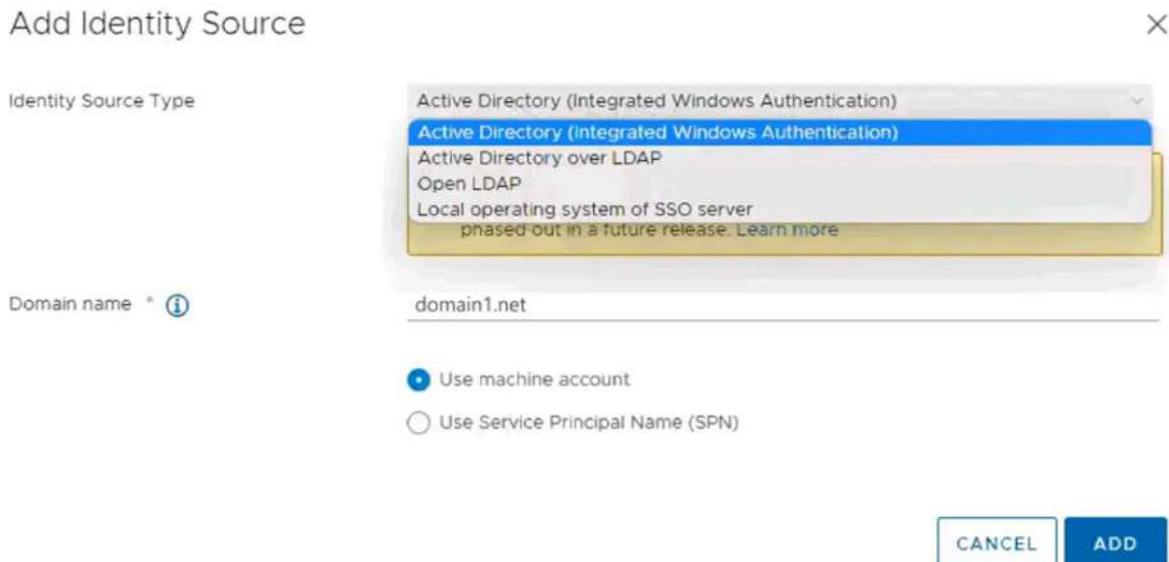
1. Go back to **Administration** and click **Configuration** under the **Single Sign-On** menu. Click the **Identity Sources** in the **Identity Provider** tab and then click the **ADD** button to add an identity source.

The screenshot shows the vSphere Client interface with the 'Administration' menu selected. Under 'Configuration', the 'Identity Provider' tab is active. In the 'Identity Sources' section, there is a table with two entries:

Name	Server URL	Type	Domain	Alias
System Domain	--	System Domain	vsphere.local	--
Local OS (Default)	--	Local OS (Default)	localos	--

A red box highlights the 'ADD' button in the top right corner of the table header area. Below the table, a message indicates '2 items'.

2. We select the **Active Directory (Integrated Windows Authentication)** option. Now that we have joined our vCenter to the domain, the **Domain name** field is automatically populated with our domain name. We can leave the **Use machine account** as the default option here. Finally, complete the configuration of the identity source and click



3. Now, under the identity source, we can see our domain. You can click **Set as default** to use this Active Directory domain by default.

Type	Embedded
Active Directory Domain	vsphere.local
Smart Card Authentication	localos
<b>Active Directory (Integrated Windows Authentication)</b>	<b>domain1.net</b>

Then you can [create Roles in vCenter](#) and assign privileges to those roles and then attach a role to an Active Directory user.

## Conclusion

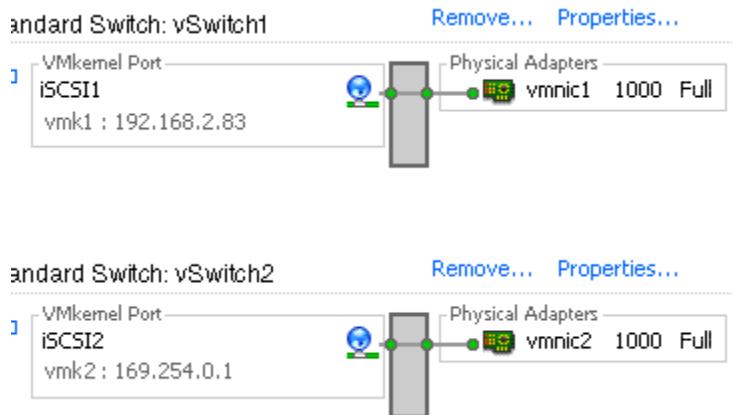
Having a centralized system for user authentication in your environment and using Active Directory for vSphere user authentication is useful in many situations. Make sure to [back up your Active Directory](#) domain controller and vCenter Server appliance regularly to avoid downtime and issues caused by the inability to authenticate users and manage the infrastructure. NAKIVO Backup & Replication is a complete data protection solution for VMware vSphere environments. Use the solution to back up your VMs and applications like Microsoft Active Directory.



## Practical No 4

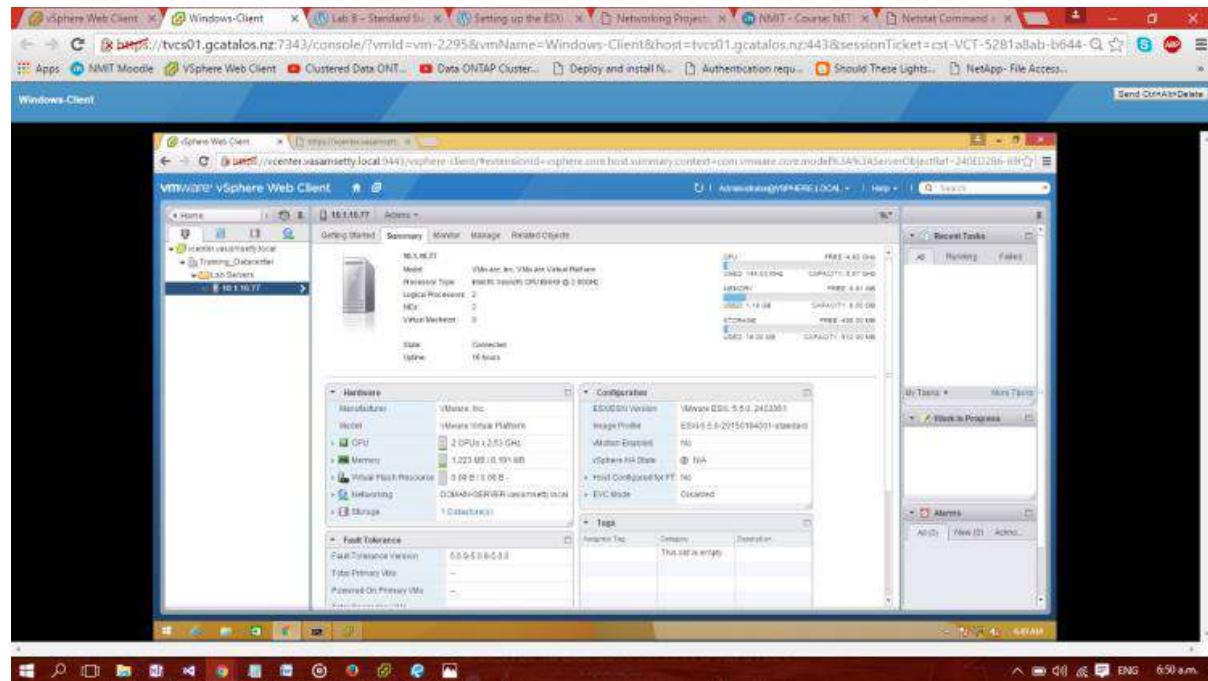
1. how to view the current standard switch Configuration
2. Create a Standard Switch with a Virtual Machine Port Group
3. Attach your Virtual Machine to a new Virtual Machine port Group

### Task-1: View the Standard Switch



In the login page provide the user credentials and login

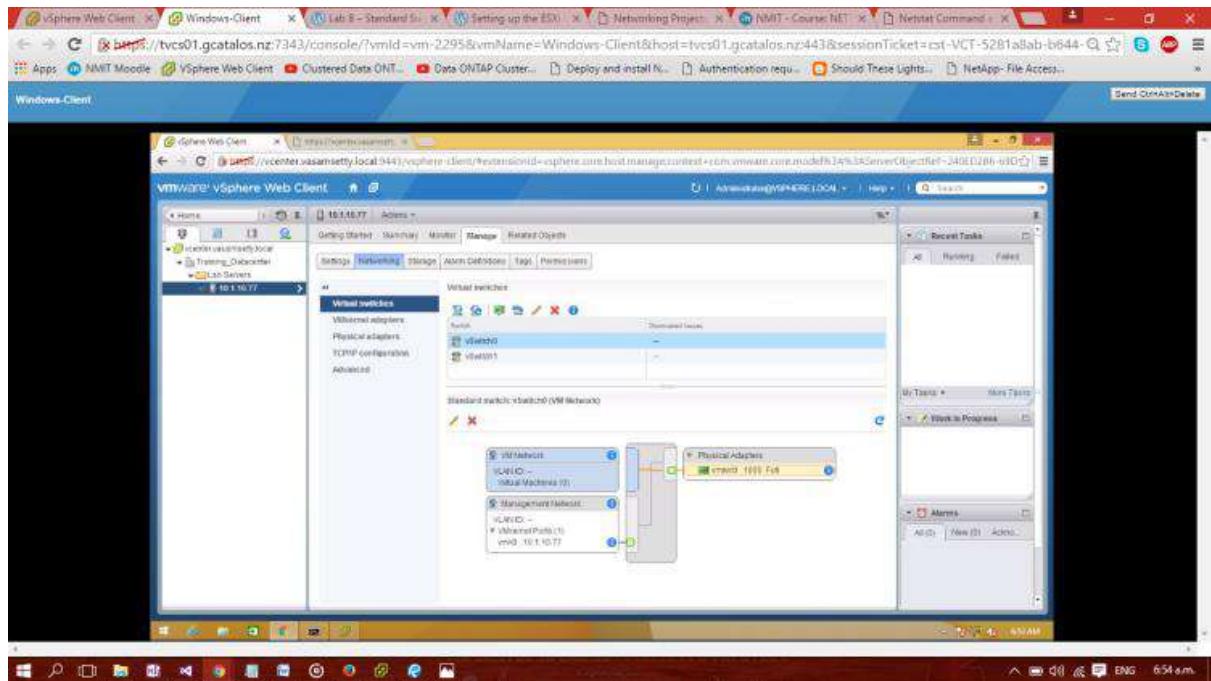
on the dashboard click on your ESXi host> you will be moved to ESXi dashboard page showing the host information like health, processor etc...



in the host dashboard click on **Manage>Networking>Virtual switches.**

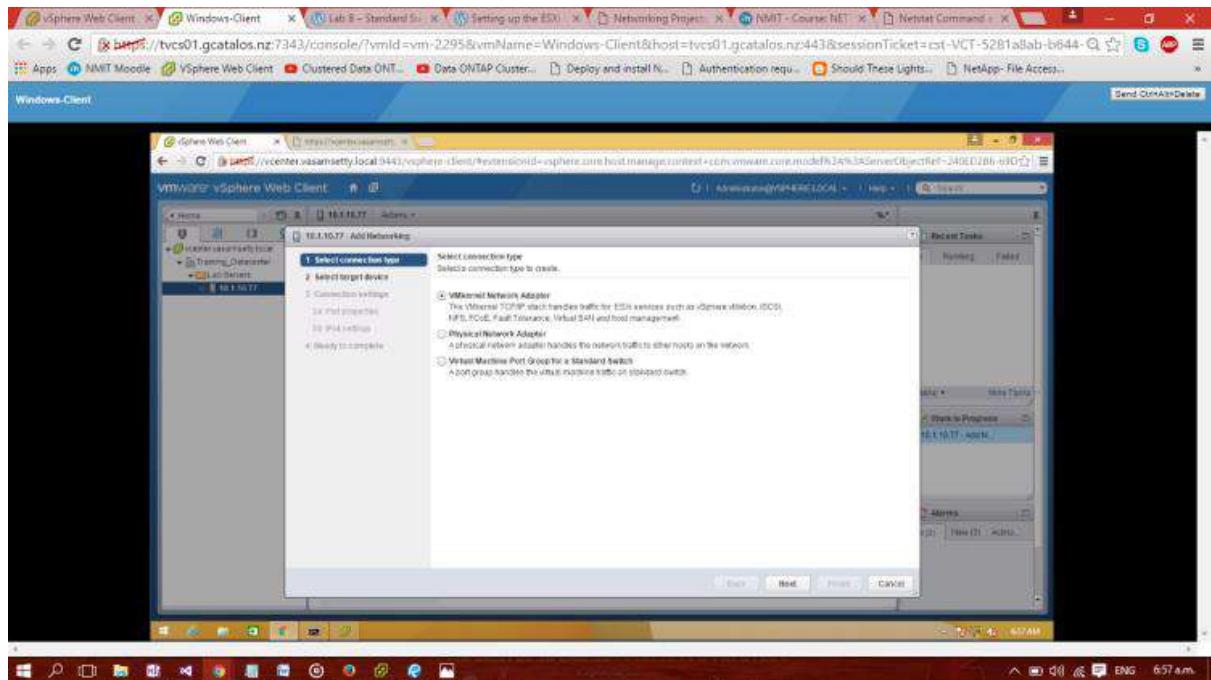
The default virtual switch is named as vSwitch0 which is connected to the physical adapted vmnic0.vswitch0 contains a VMkernel port named Management Network and a

virtual machine port group named VM Network. A virtual machine is connected to VM Network.



## Task-2: Creating a Standard Switch

to create a standard switch goto the same path as above in the Task-1. Click on globe shaped icon with green plus sign or the first icon on the left and give the following configuration details



Connection Type : Virtual Machine Port Group for a Standard Switch

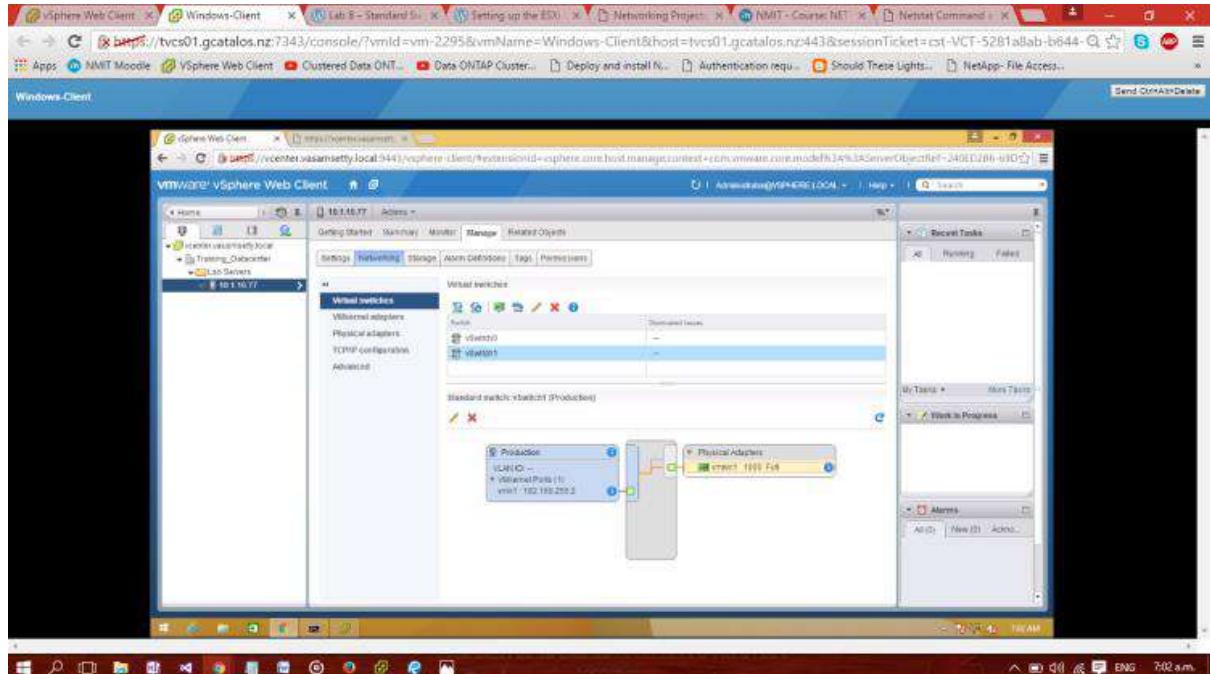
Target Device : New Standard Switch

Create a Standard Switch: click on + and add a vmnics which are available to the network and click ok.

Connection settings: Default

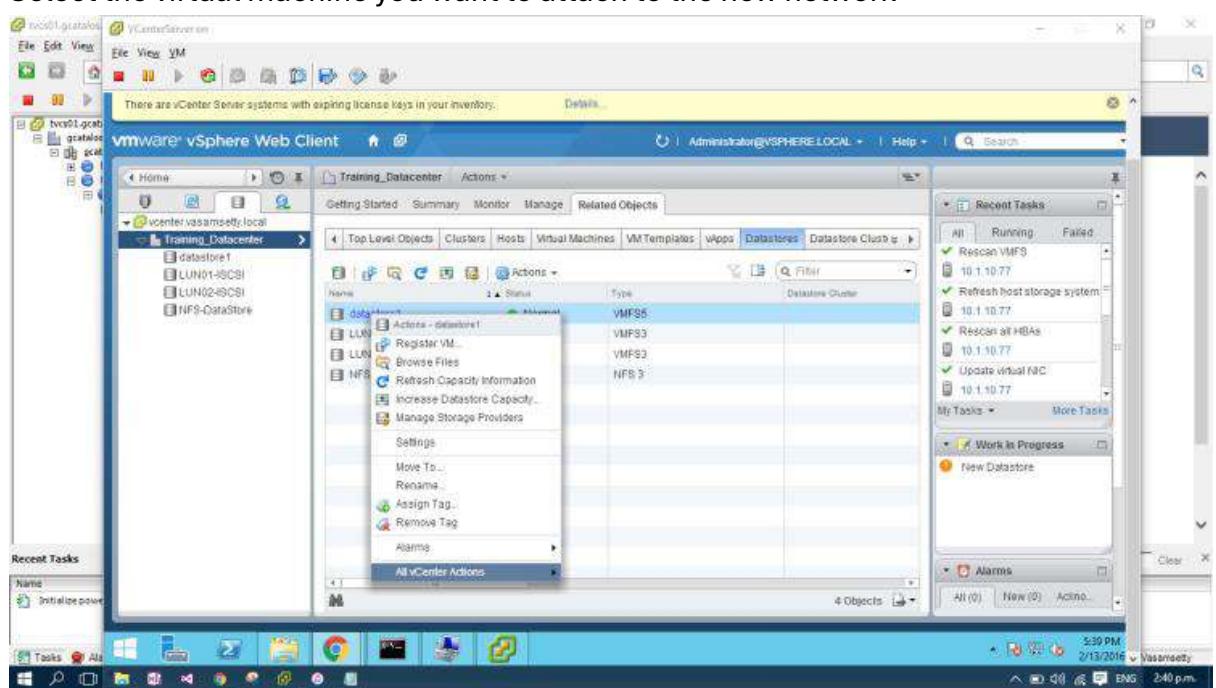
Name tag: vSwitch1

Once the switch is created you can see the switch details in the **Hosts>Manage>Networking>VirtualSwitches**

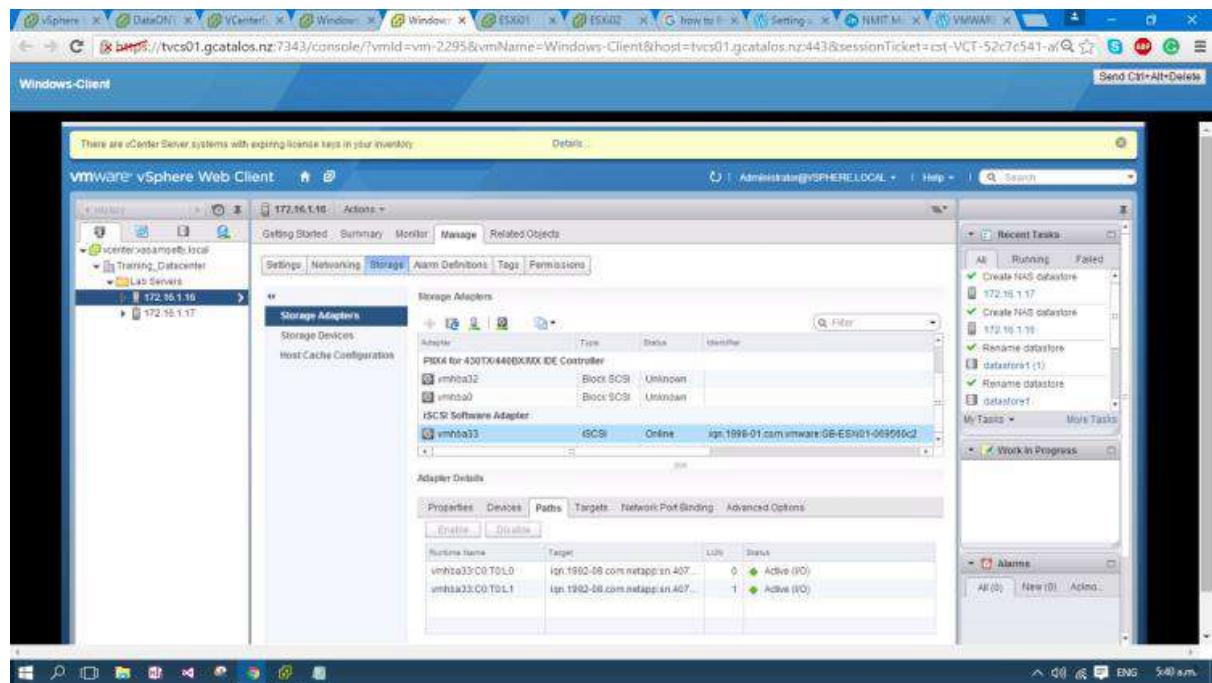


### Task-3: Attach your Virtual Machine to a New Virtual port Group

Select the virtual machine you want to attach to the new network



In the Virtual Hardware tab, click on the Add New Device button.



## Practical No 5

### Accessing iSCSI Storage

In this lab, I was required to do the following tasks:

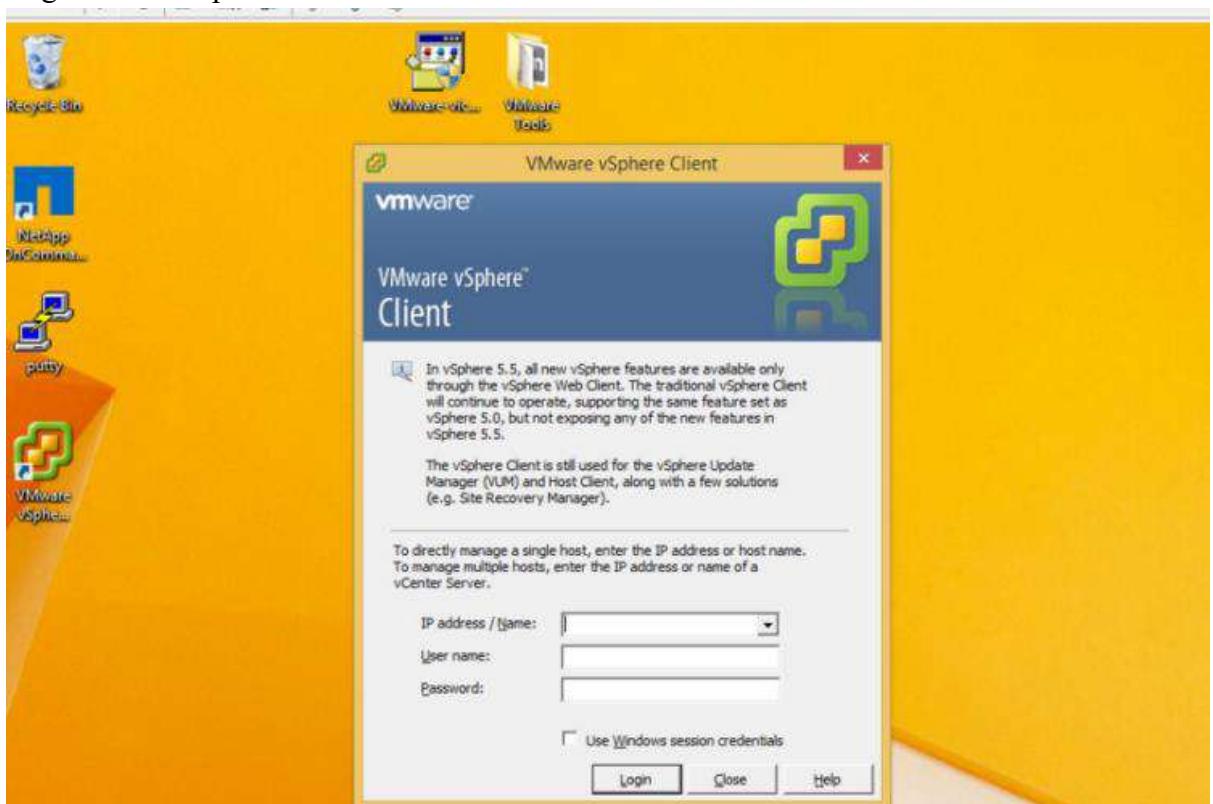
- validate an existing ESXi host's iSCSI configuration in VMware
- Add a VMkernel Port Group to a Standard Switch
- Configure the iSCSI Software Adapter
- connect the iSCSI software adapter to storage

validate an existing ESXi host's iSCSI configuration in VMware, follow these steps:

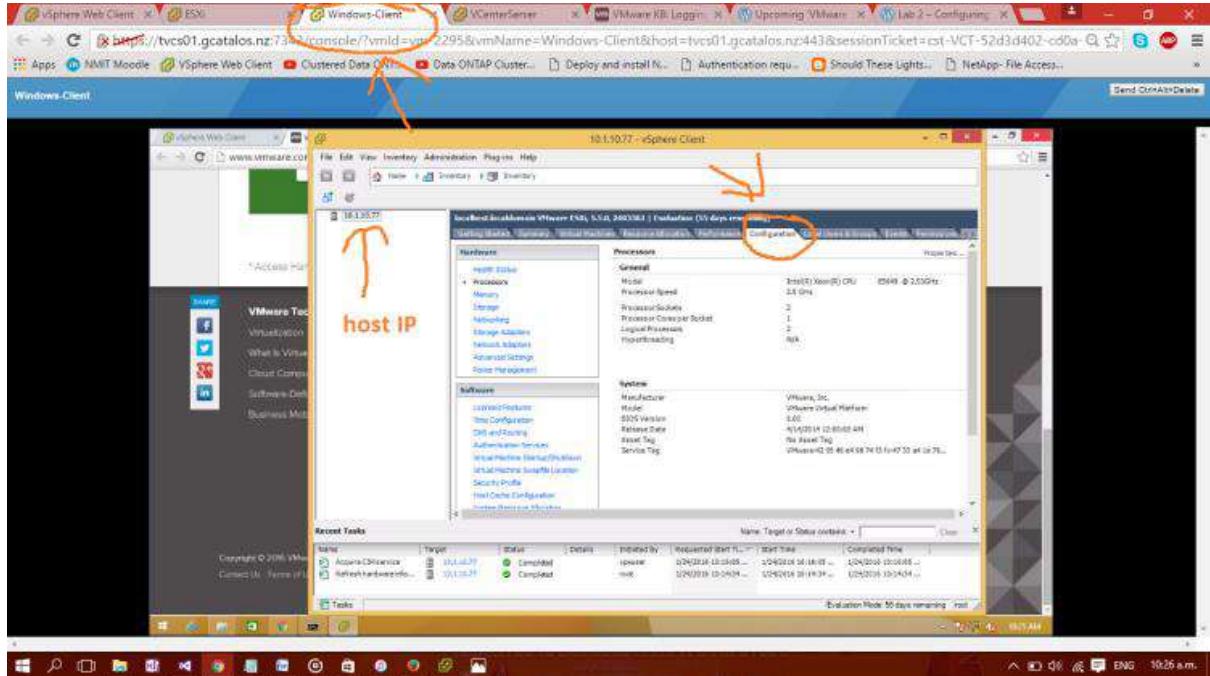
Using the vSphere Web Client



1. Log in to the vSphere Web Client.



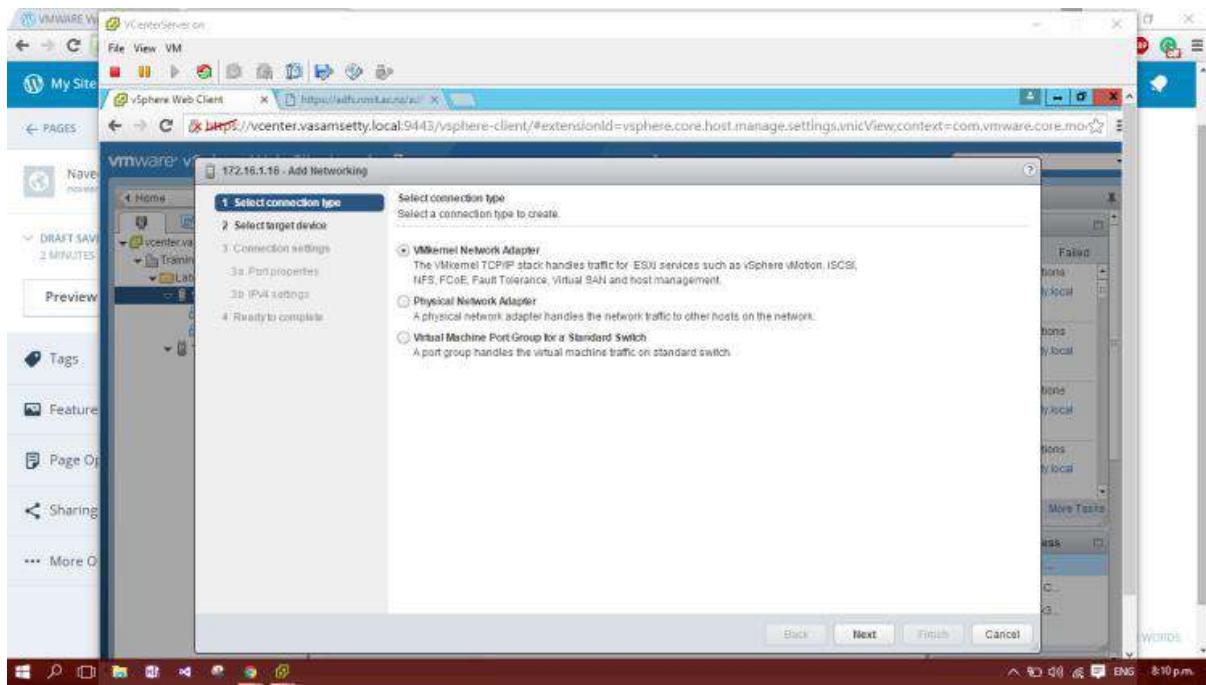
2. Navigate to the Hosts and Clusters view.
3. Select the ESXi host with the iSCSI configuration you want to validate.
4. Click on the Configure tab.



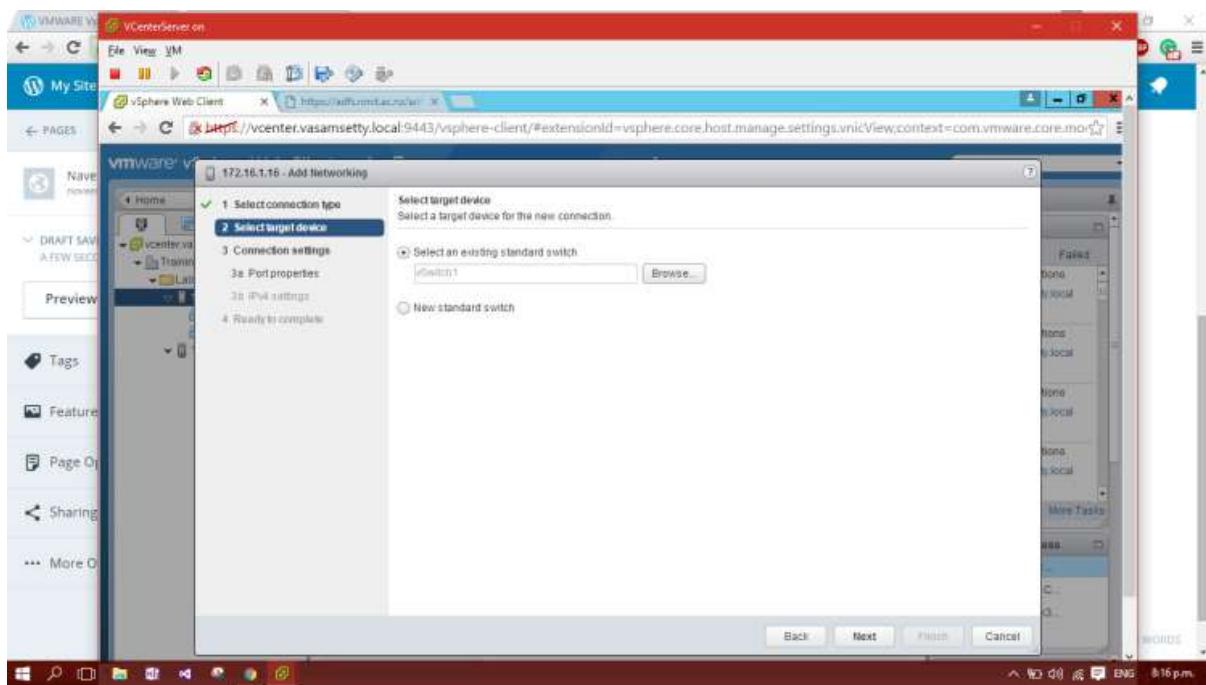
5. Click on Storage.
6. Select Storage Adapters.
7. Click on the iSCSI adapter.
8. Verify the iSCSI settings, including:
  - iSCSI initiator name
  - iSCSI target IP address and port
  - CHAP authentication settings (if enabled)
9. Click on Targets to verify the connected iSCSI targets.
10. Verify the iSCSI LUNs (Logical Unit Numbers) are visible and accessible.

#### Adding a VMkernal Port Group to a Standard Switch

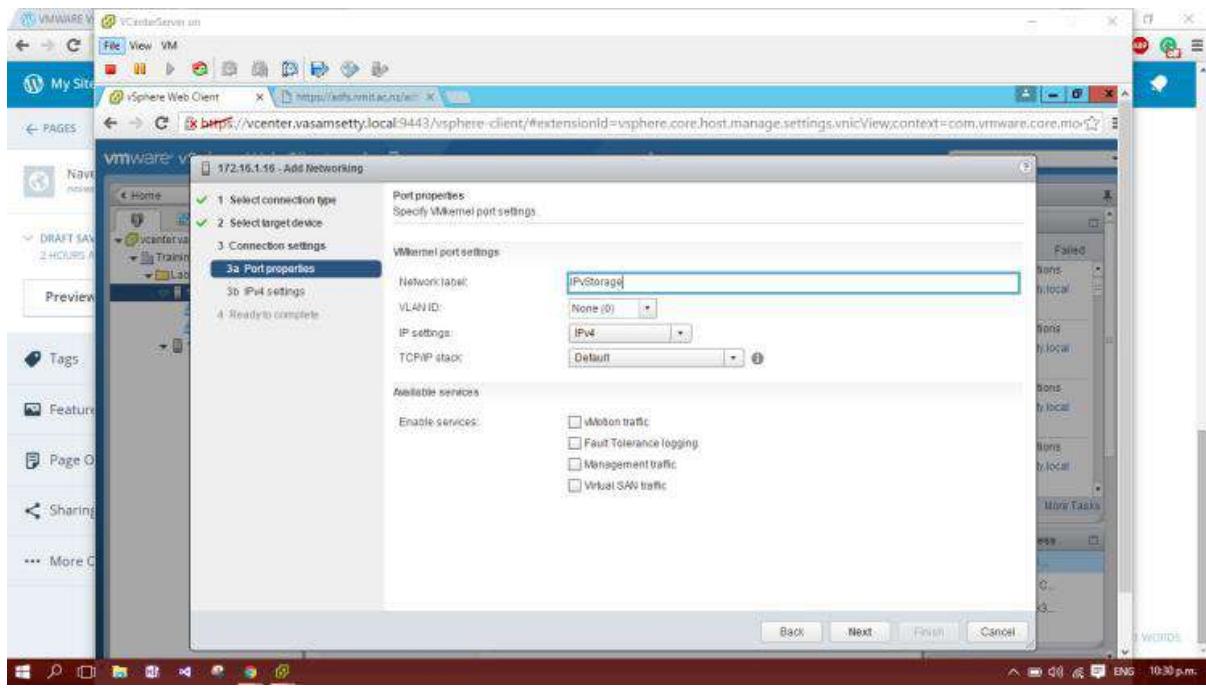
In the previous post I just created two Standard switches. And now I added VMKernal port to newly created Standard switches by navigating to vCenter -> Hosts and Clusters', selected my first ESXi host, clicked the 'Manage' tab, clicked the 'Networking' tab, clicked on 'Virtual switches', and finally, selected 'vSwitch1'.



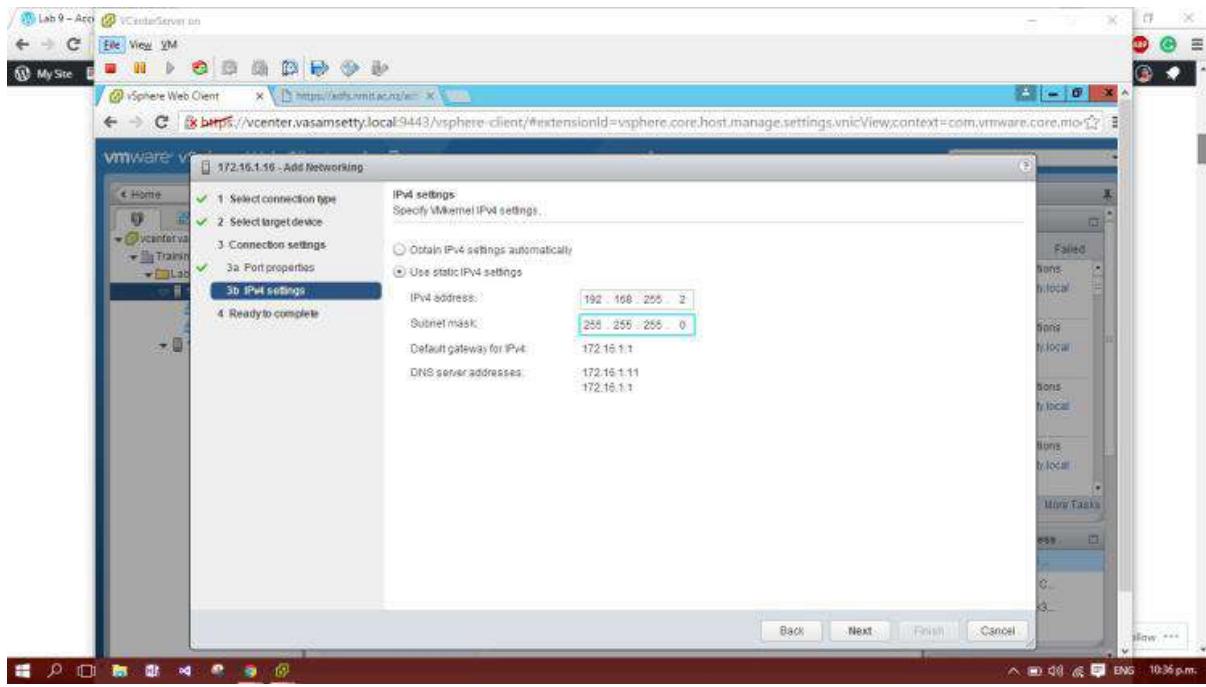
I continued the execution by clicking next. In the next step i choose Vswitch1 switch I did not change anything here after and just stuck with the defaults.

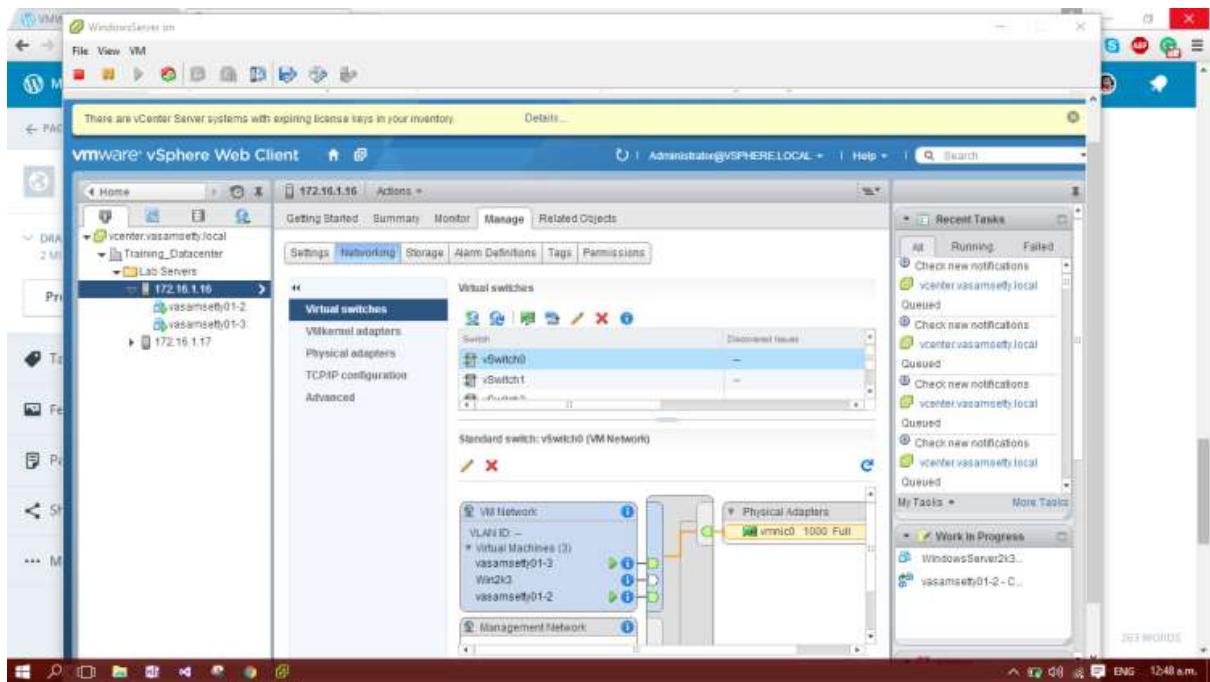


For the network label, I entered in IP Storage'. I left the remaining settings at their defaults.



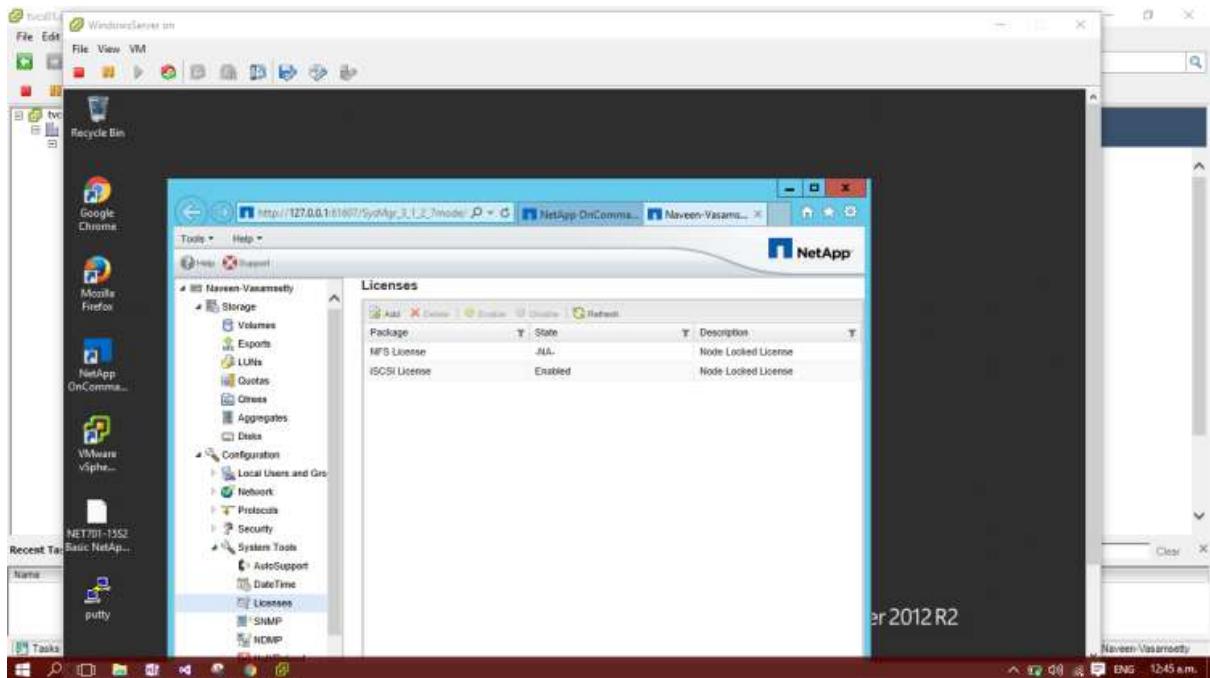
Click on next to give the IPv4 settings. In the IPv4, I had given an IP address based on my network diagram .This IP may use as a switch two connect virtual machines that i am gonna create So, finally after creation of VMkernal adapter I observe the following switches port details.





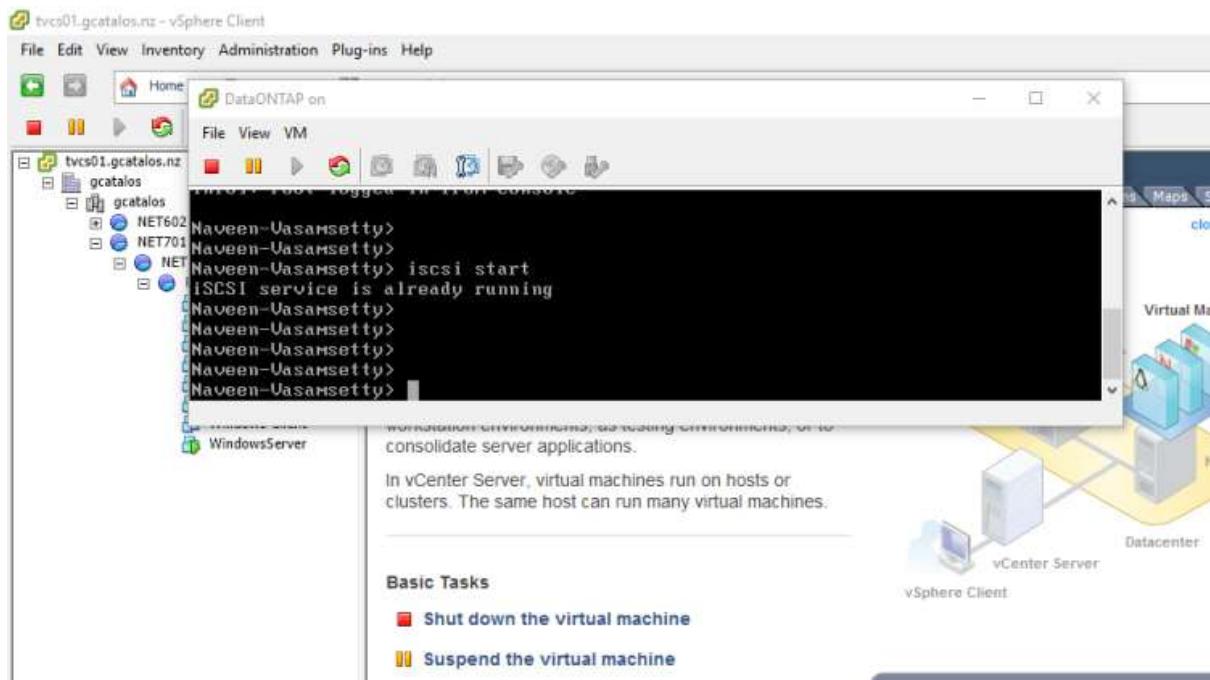
Pre-work before configuring the iSCSI Adapter: Before making a connection to my vsim and configuring the iSCSI software adapter in this task. I need to enable my iSCSI services on my VSIM. For this I had given an iSCSI license key and enabled the services using the command prompt command ISCSI START.

Configuring my VSIM is done through on NetApp Manager. I just logged into one of my client system opened the NetApp manager and given a NFS and iSCSI license file in it.



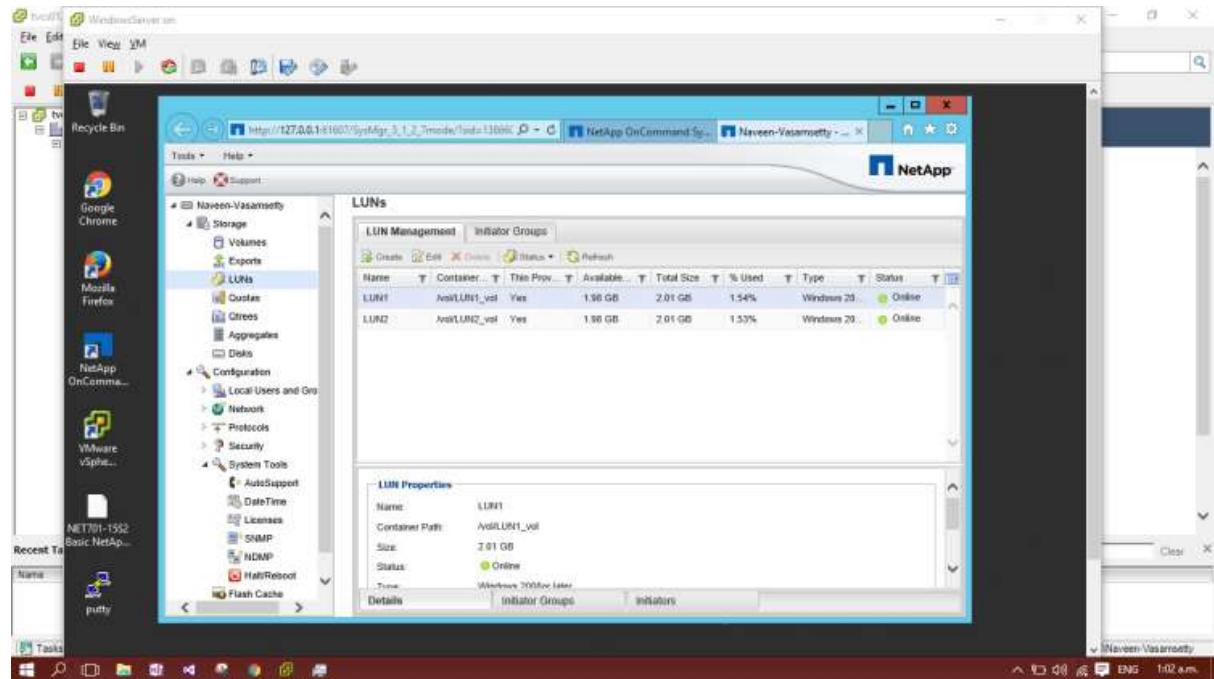
and for starting the iSCSI services i just opened my VSIM VM and entered the command.

**ISCSI START**

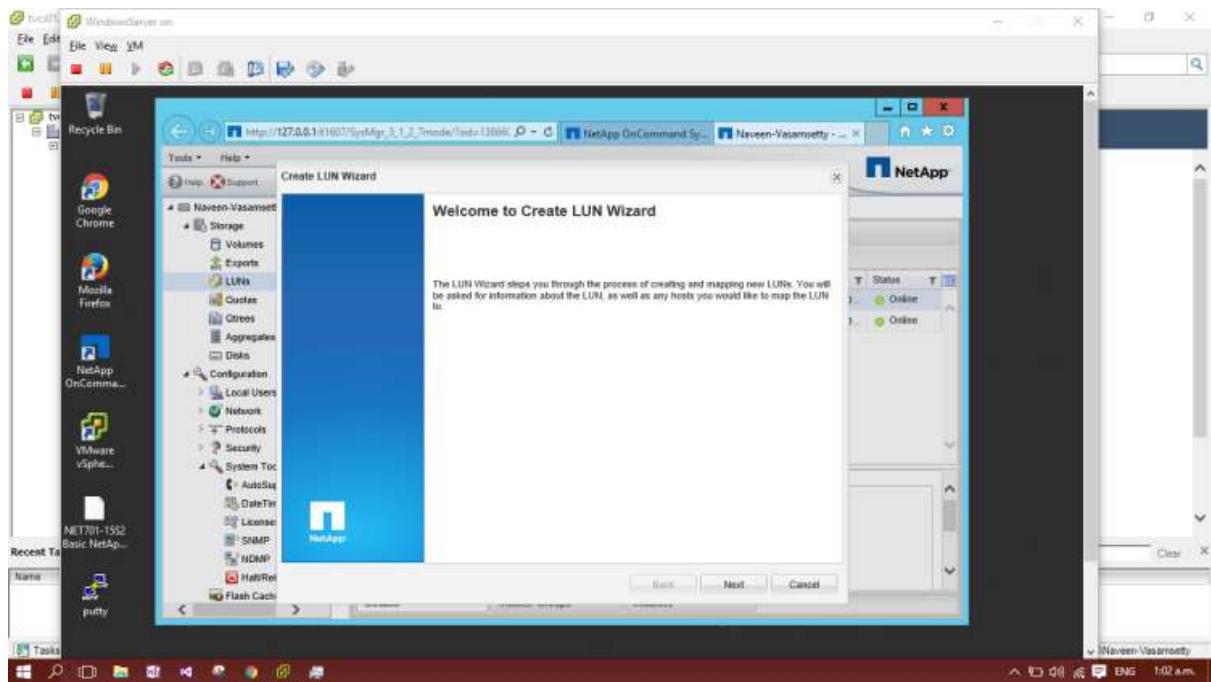


This made me an opportunity to create some luns in my VSIM it will be very useful in future tasks. Creating a LUN is easy through NetApp Manager. I opened NetApp manager session and navigated to the LUNS and created two luns namely LUN1 and LUN2. I can't add the initiator group at this time because I don't have the initiator id.

I just logged into my NetApp manager and clicked on LUNS. As you can able to see they are already build luns LUN1 and LUN2. I am gonna create a new lun named LUN3 in this session.

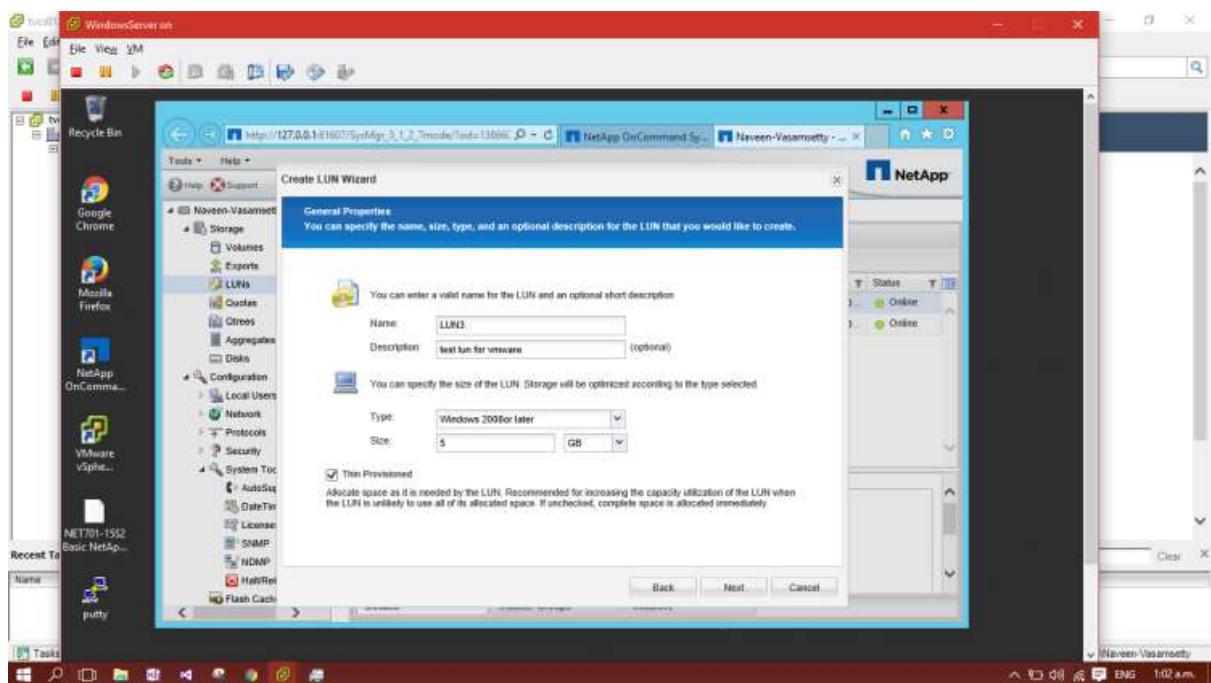


Click on Create- you can see a pop-up window like below

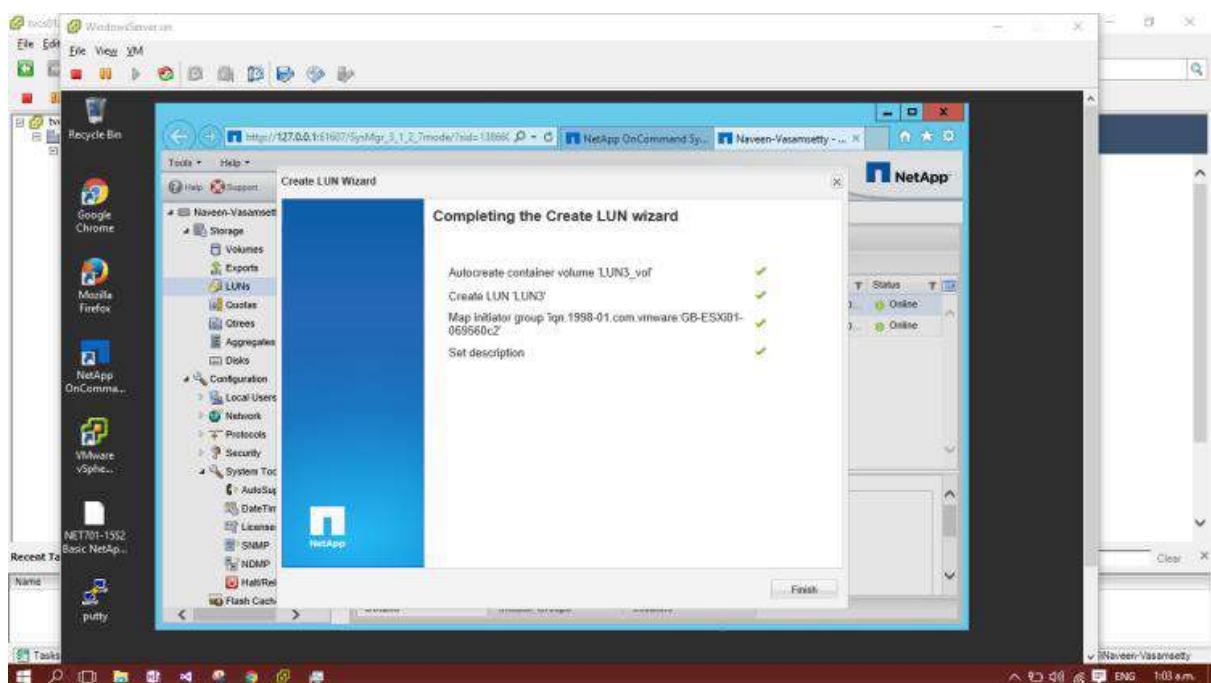
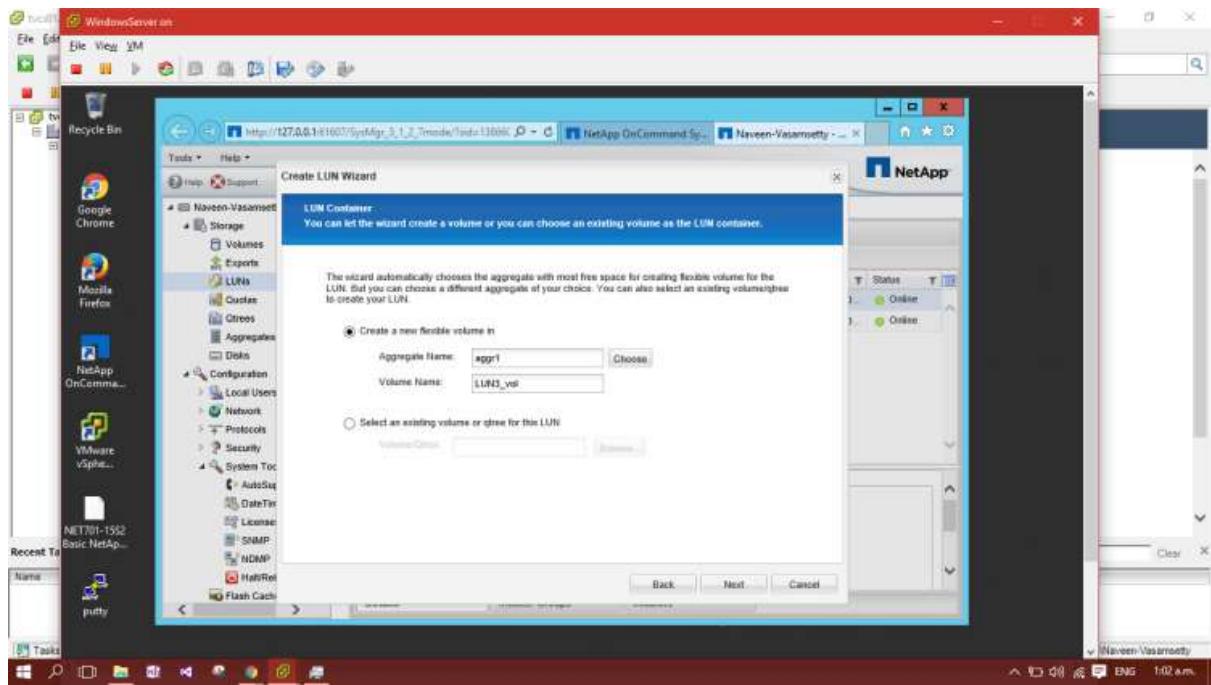


I had given the LUN name and given a short description about the lun.

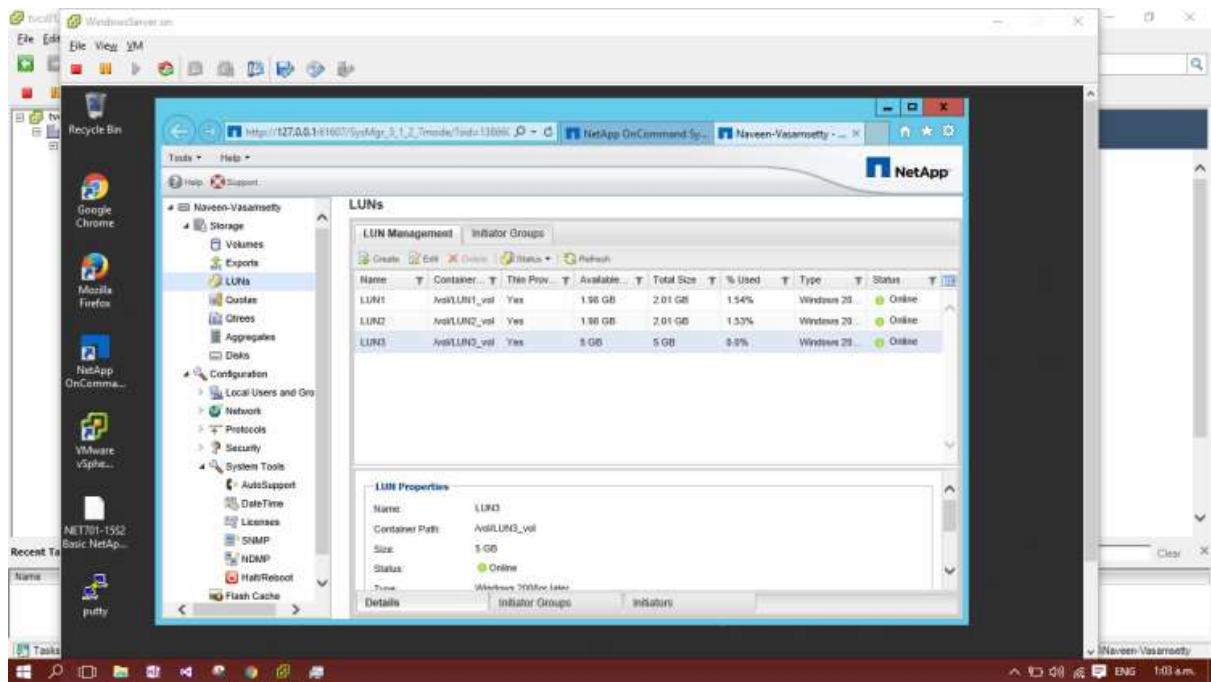
I specified the type as Windows 2008 and later and given a size of 5 GB.



it specifies automatically the path of volume and aggregate

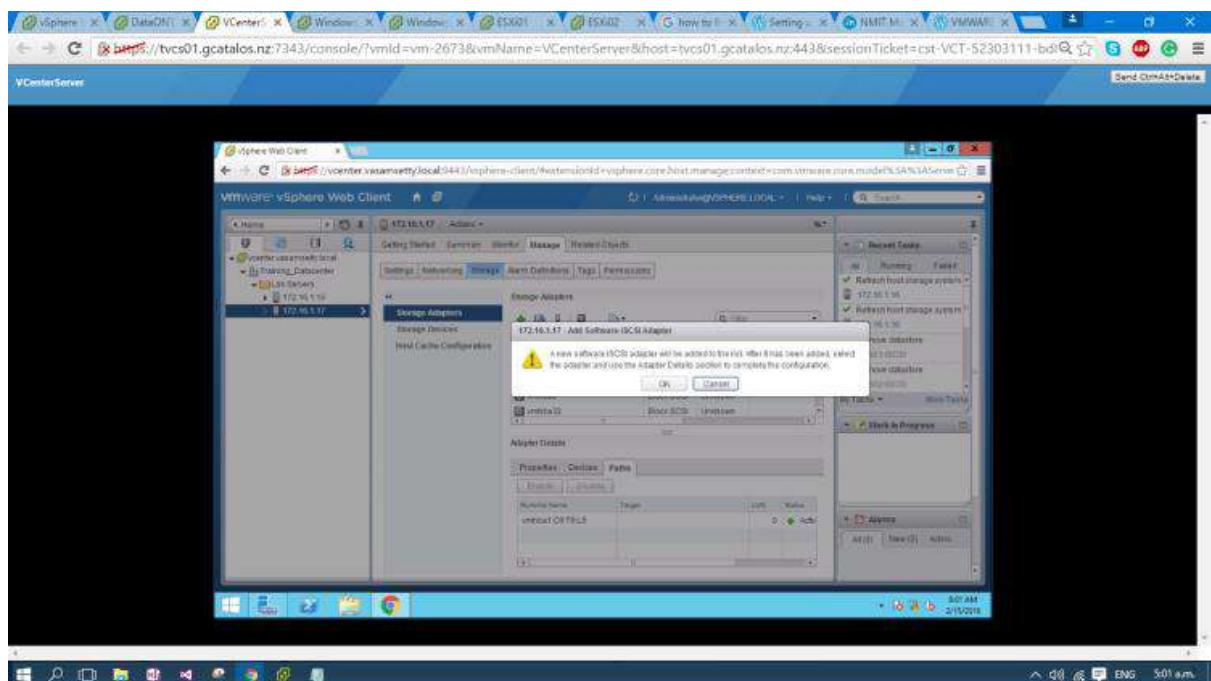


After a successful attempt I can see my new LUN created

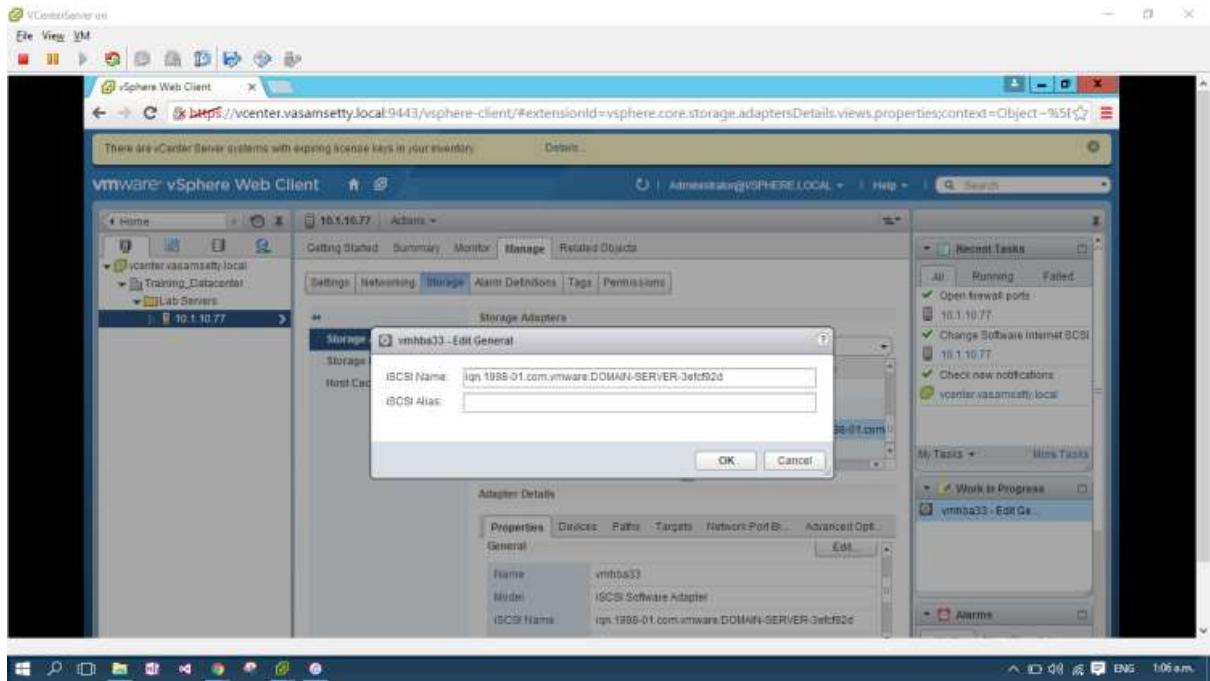


## Configure the iSCSI Software Adapter

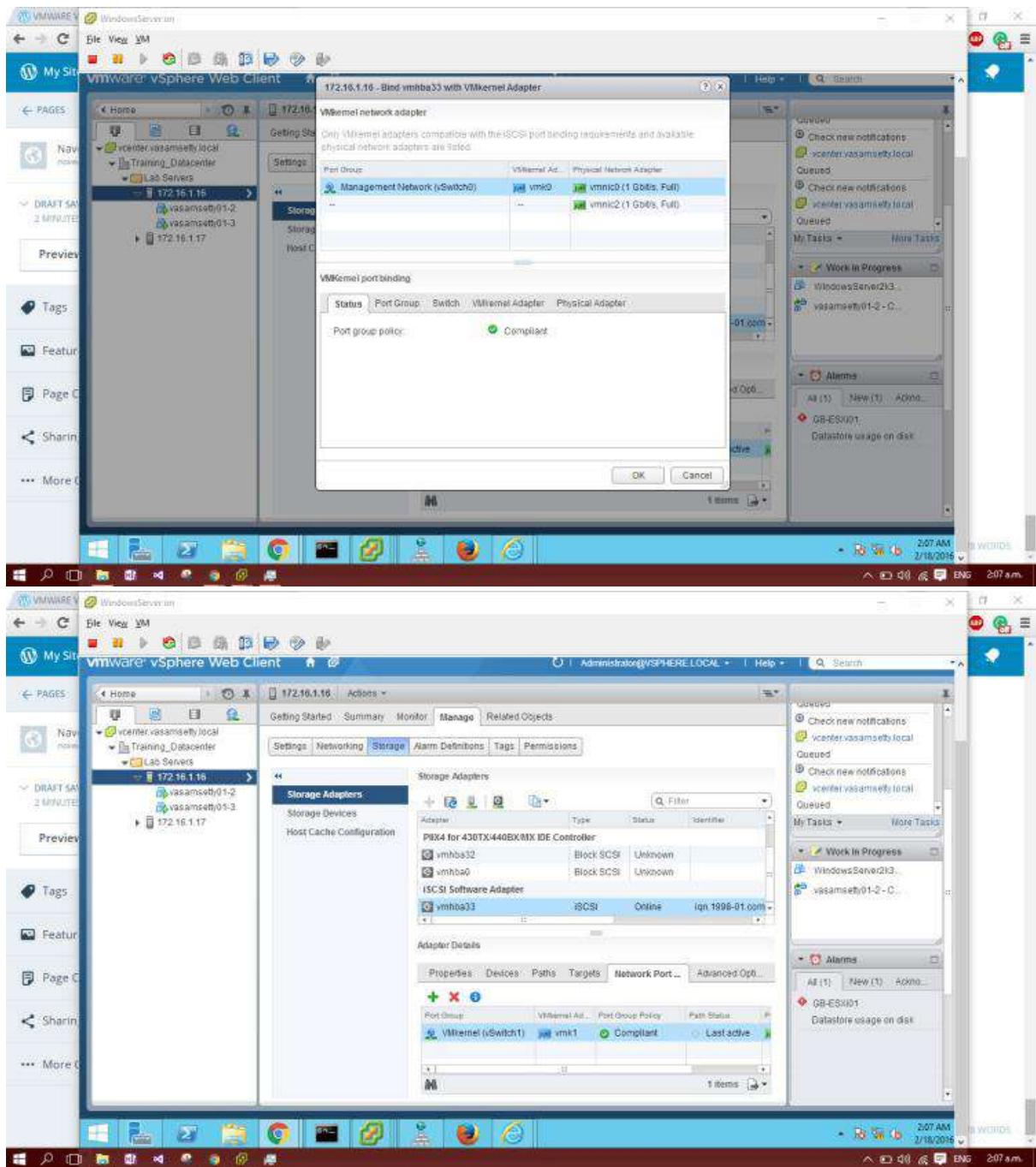
To configure my iSCSI adapter, I logged into my web client application through my Administrator@vsphere.local username and navigated to vCenter -> Hosts and Clusters', selected my first ESXi host, clicked the 'Manage' tab, clicked the 'Storage' tab, clicked on 'Storage Adapters', and finally, selected '+' sign to add a new iSCSI adapter.



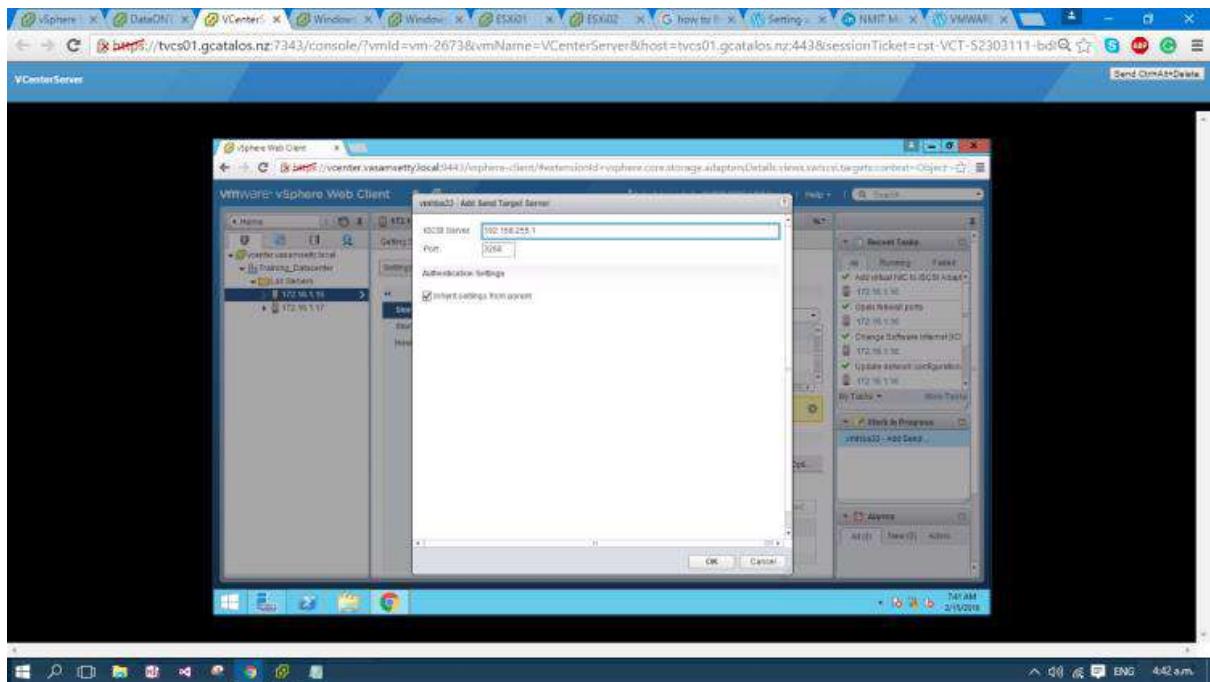
once you had successfully done in creating the iSCSI adapter you can able to see their properties, targets and network binding groups. I just gone to properties and clicked on edit button to check the iSCSI name.



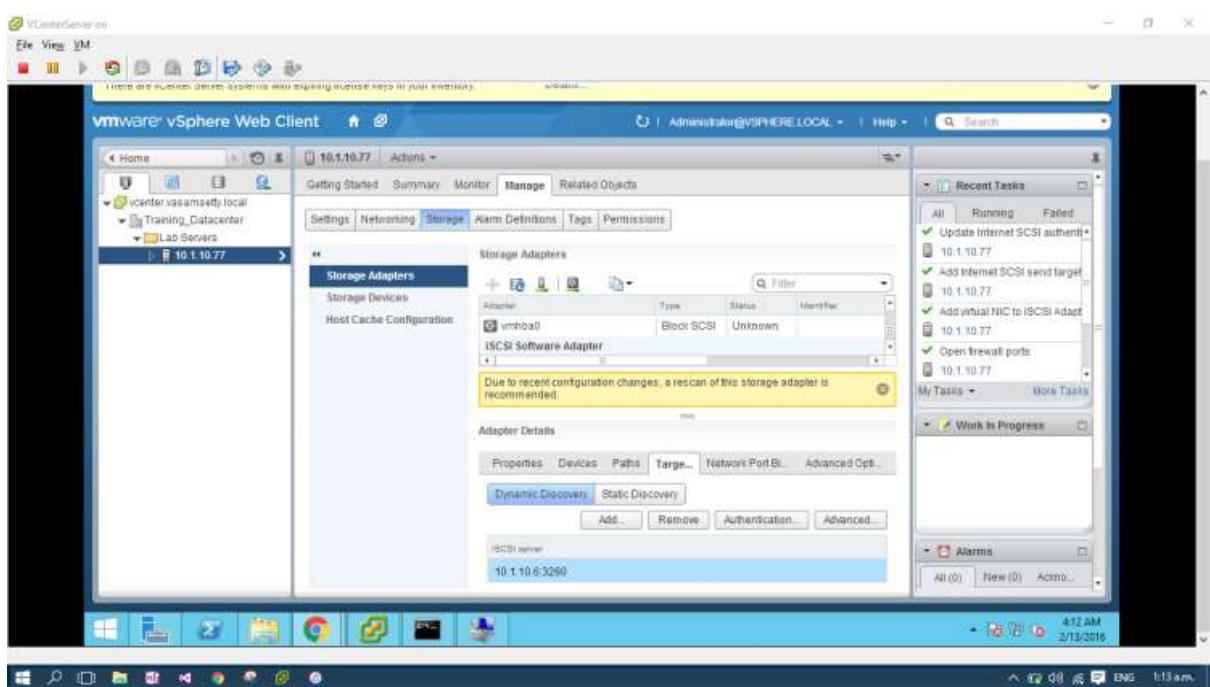
After checking that, I navigated to the ‘Network Port Binding’ tab and clicked on the little green ‘+’ symbol to add a new binding. I selected the VMkernel kernel adapter that I created earlier and clicked ‘OK’.



I then navigated to the ‘Targets’ tab, ensured ‘Dynamic Discovery’ was selected, and clicked ‘Add’. This prompted me to add in the IP address for my vSIM. I added in the IP of the e0b NIC that I created earlier on and left the port value as 3260.

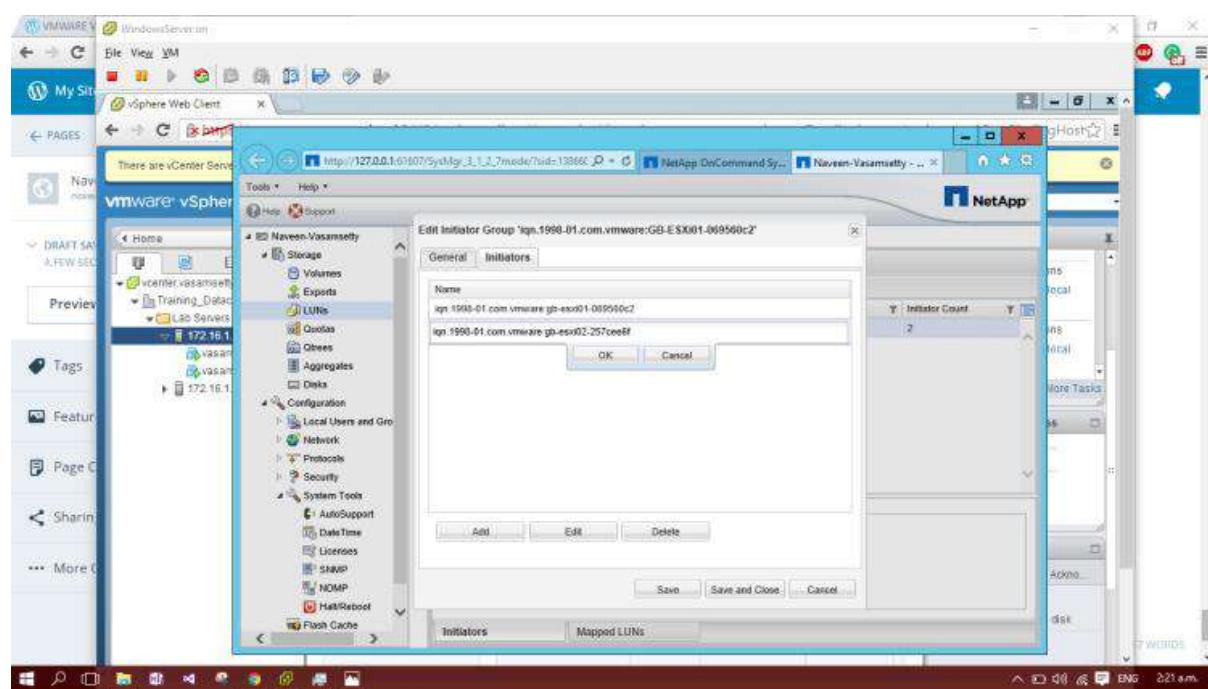
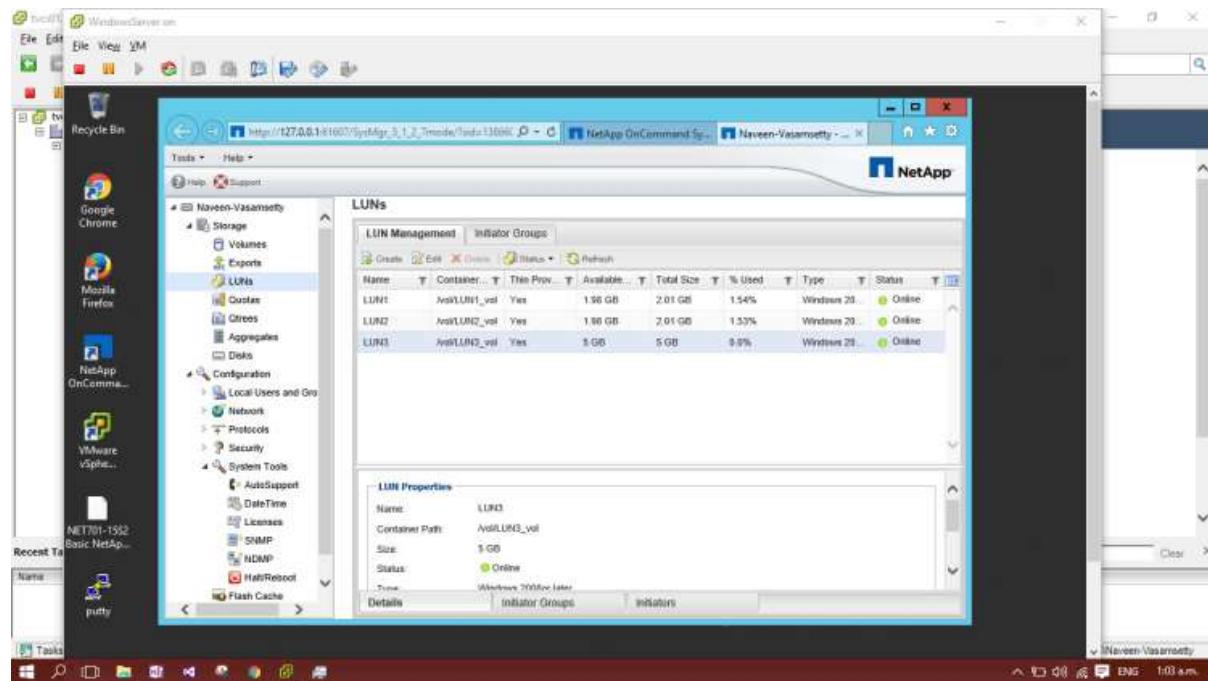


After a brief delay, the target appeared in the targets list as seen below:

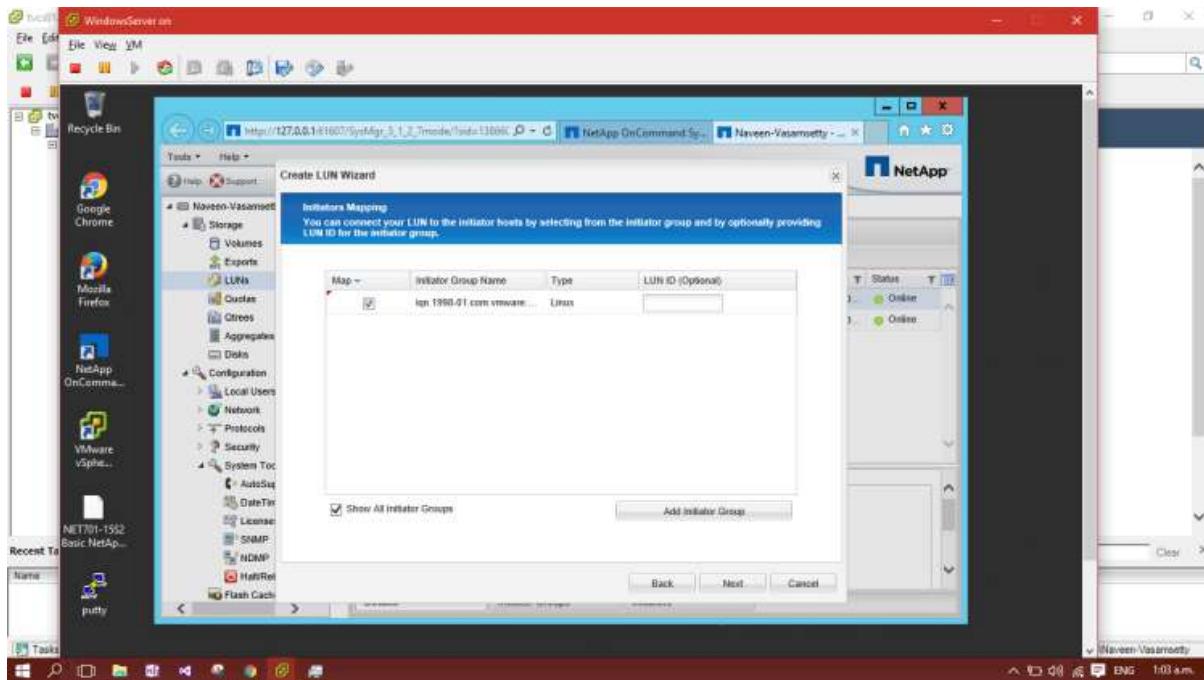


To test that everything is working as it should, I clicked on the ‘Rescan adapter’ icon and waited for the scan to finish. I then selected the iSCSI adapter and navigated to the ‘Paths’ tab, but unfortunately, no information was displayed. Its because my initiator is not attached to my created luns. To do so i copied my EXSi iSCSI adapter name to the dashboard and pasted it to my initiator group.

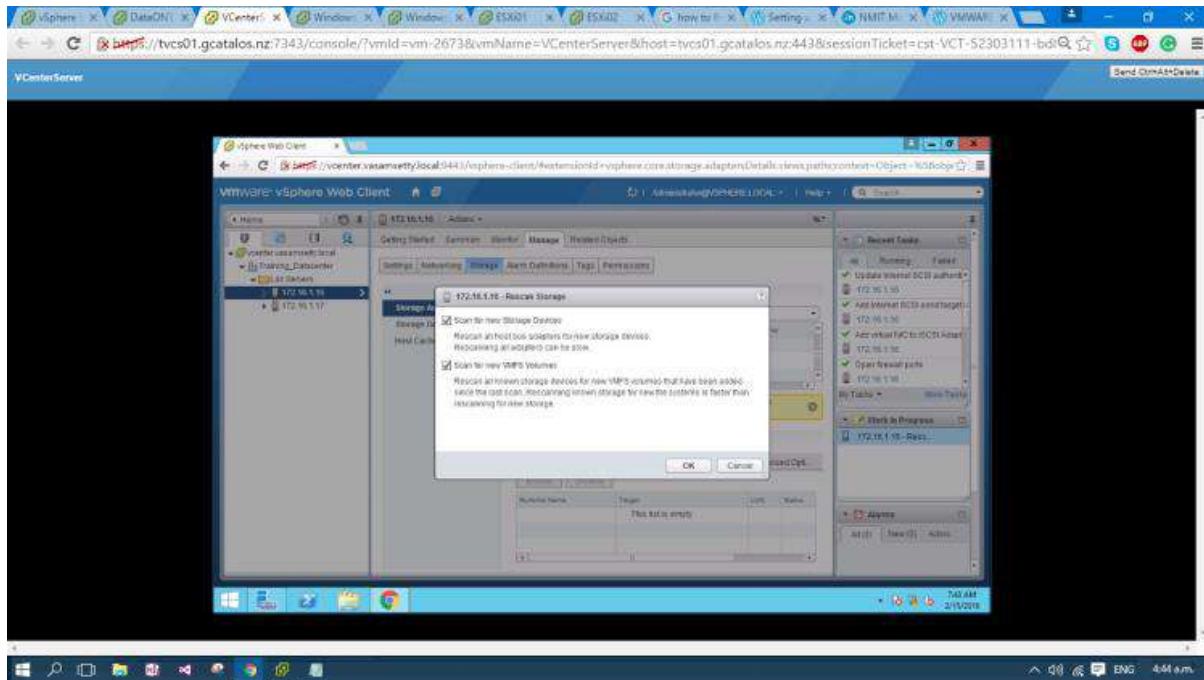
Open >NetApp Manager>LUN'S>Initiator Groups>Edit

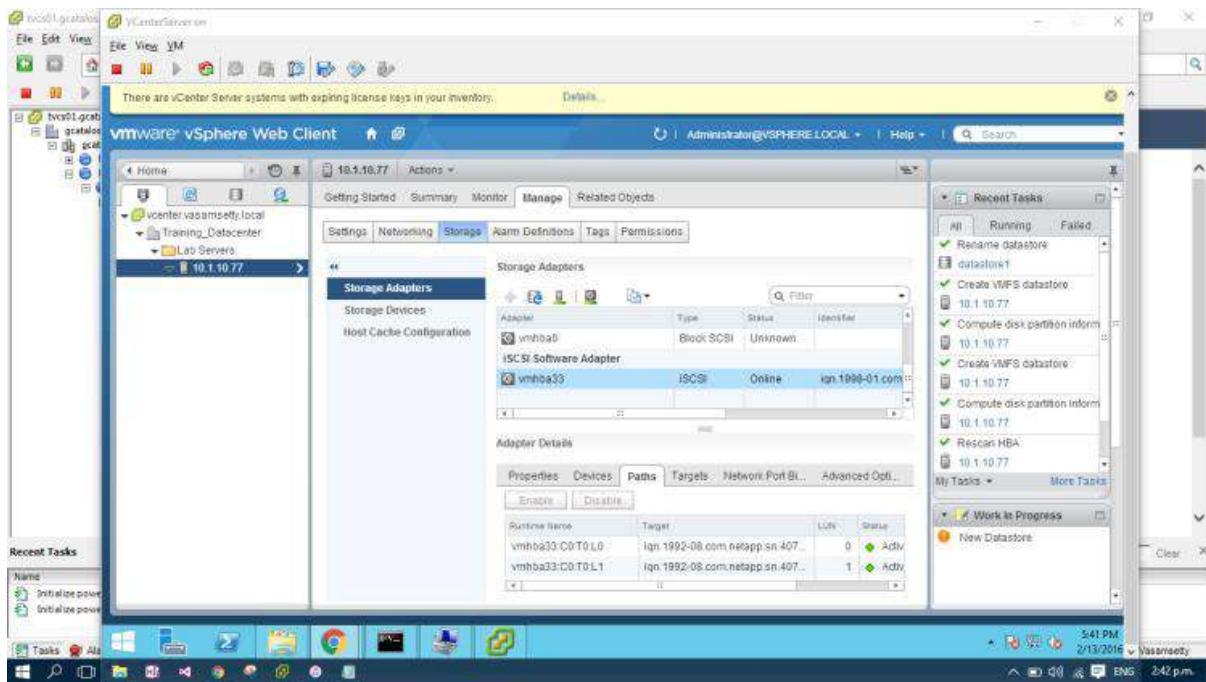


I just mapped my initiator group to my luns

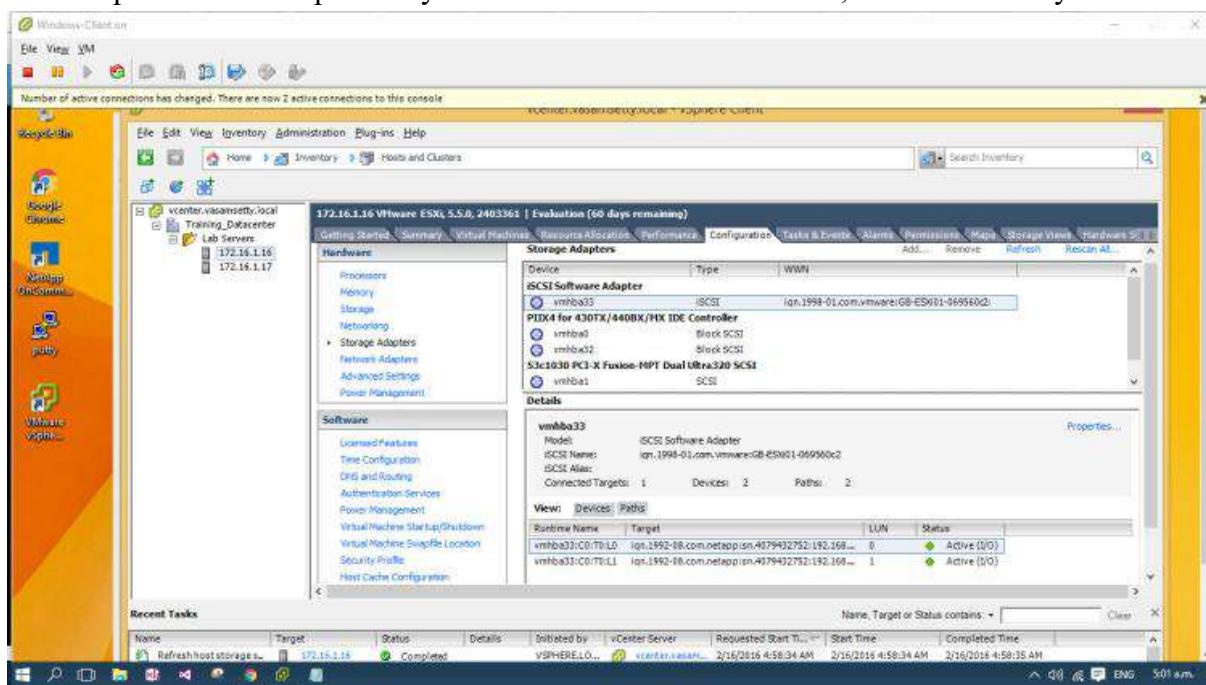


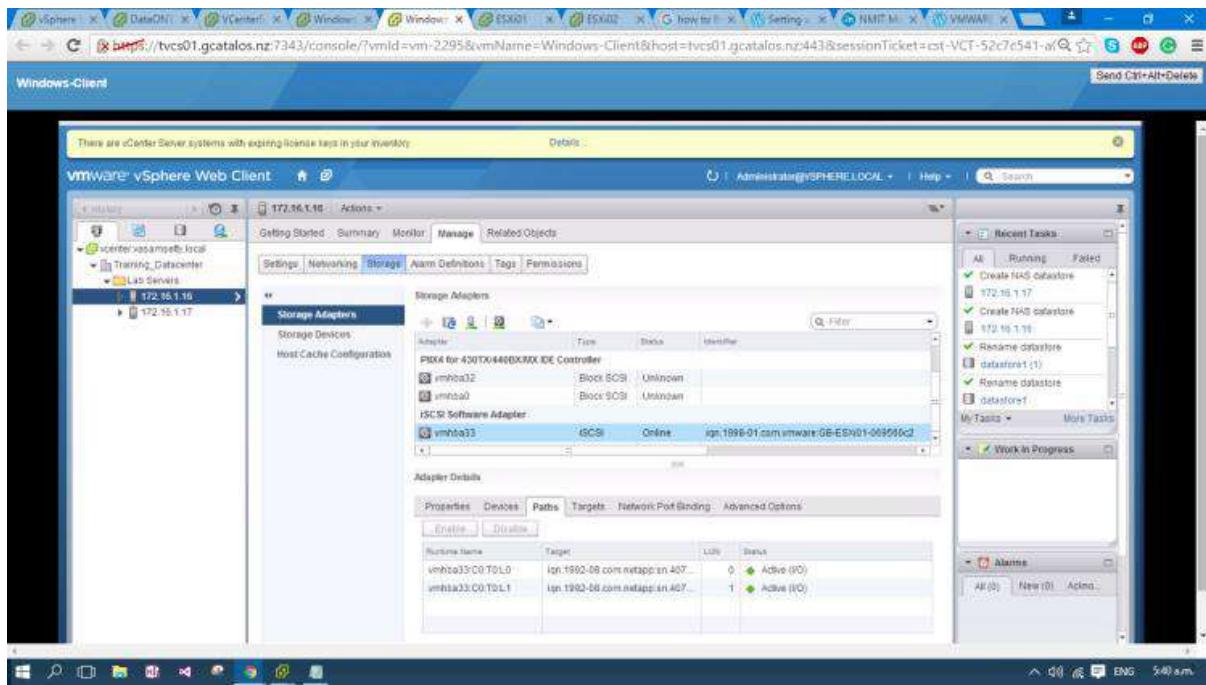
Once again, I ‘Rescan the Storage Adapters’ and found that my luns are bieng showed in the Paths tab





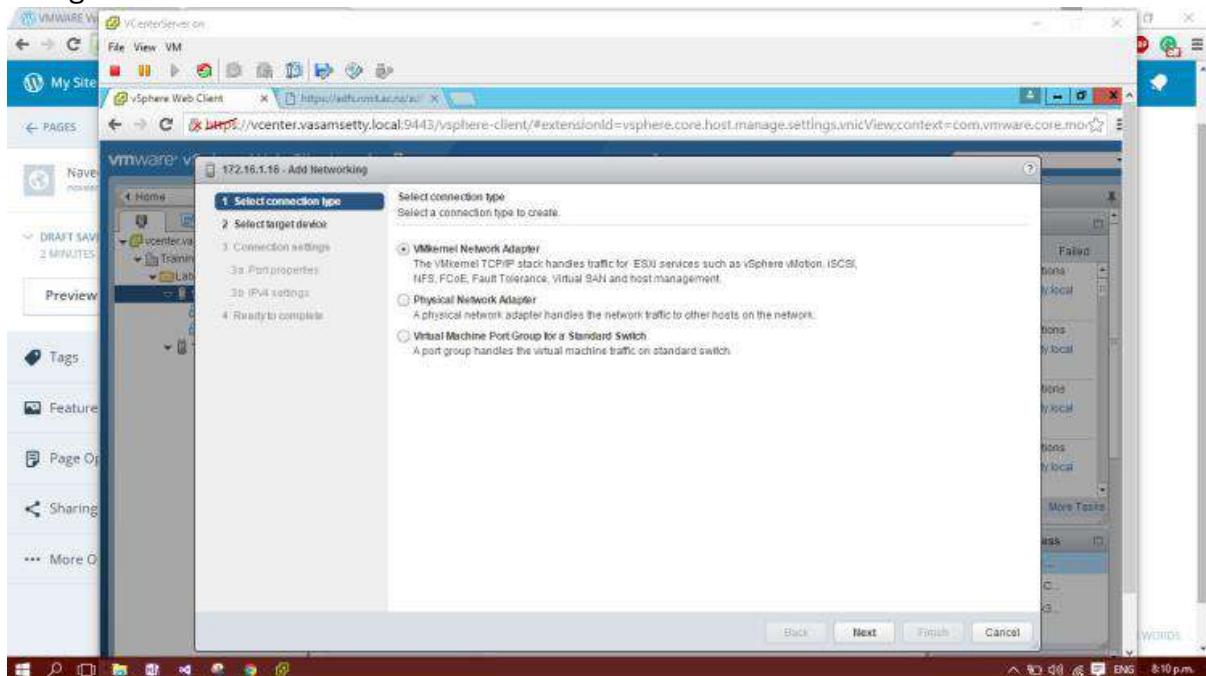
I then repeated these steps for my second ESXi host so that it too, has access to my vSIM.





## Using the vSphere Web Client

Navigate to the Hosts and Clusters view.



Select the ESXi host where you want to connect the iSCSI software adapter to storage.

Click on the Configure tab.

Click on Storage.

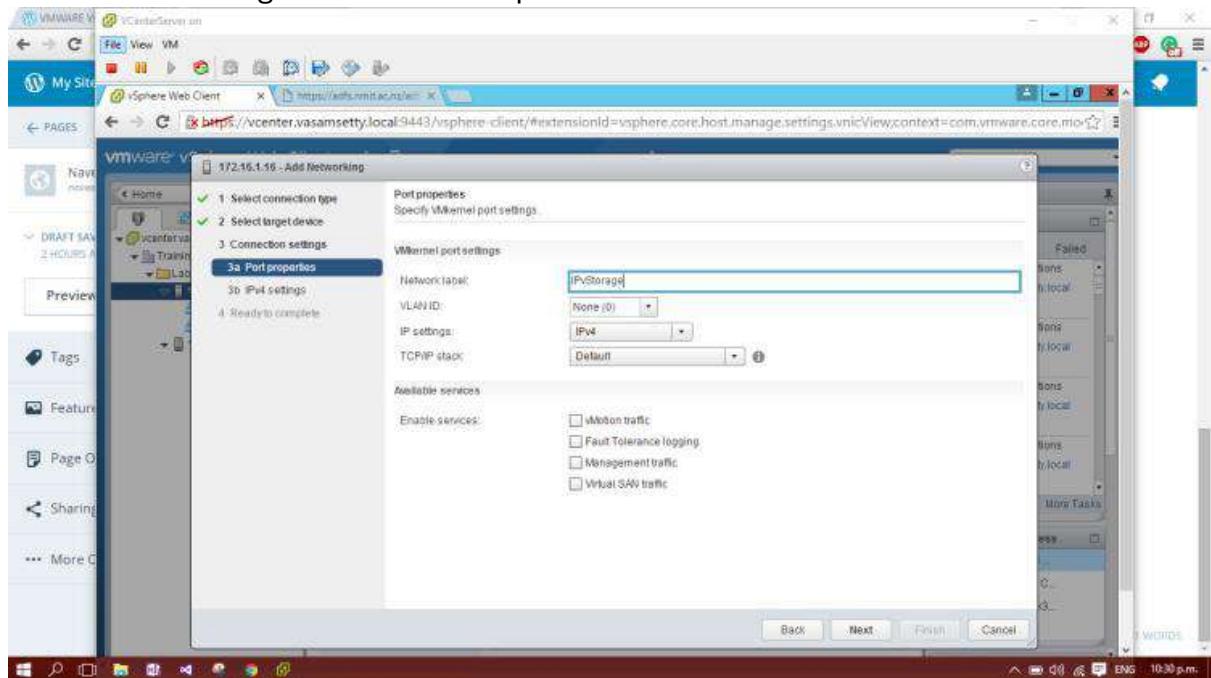
Select Storage Adapters.

Click on the iSCSI software adapter you created earlier.

Click on the Targets tab.

Click on the Add Target button.

Enter the iSCSI target IP address and port.

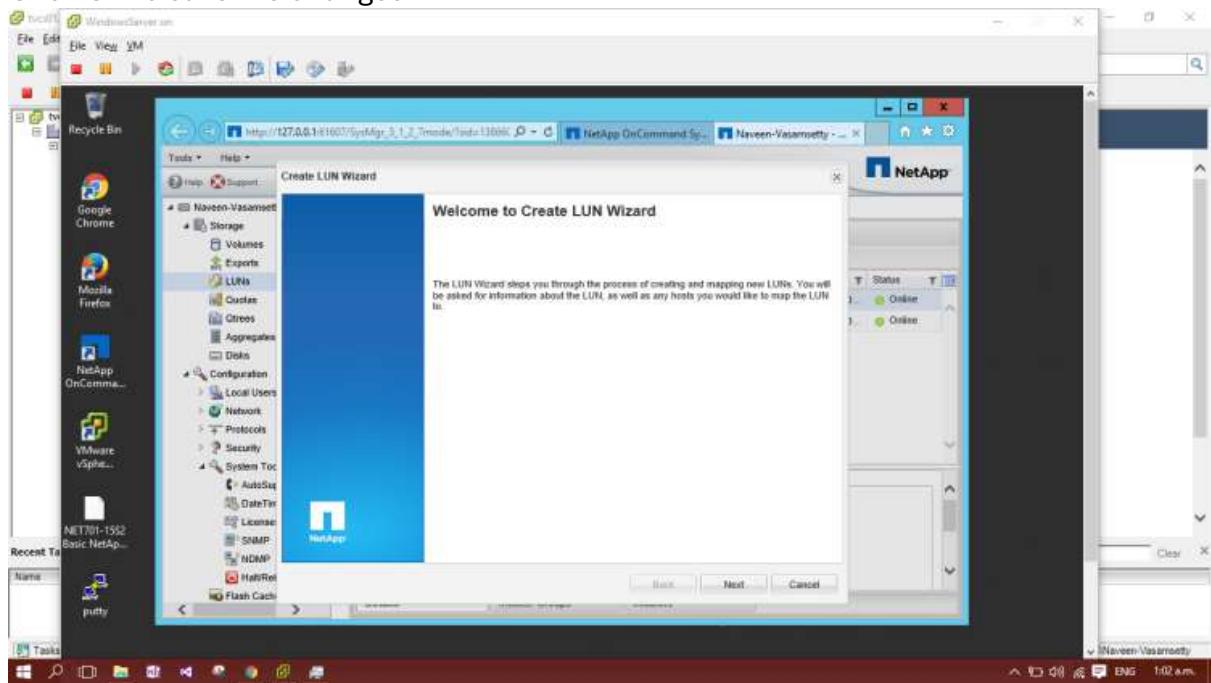


Click OK to add the target.

Click on the Dynamic Discovery tab.

Select the Enable dynamic discovery checkbox.

Click OK to save the changes.



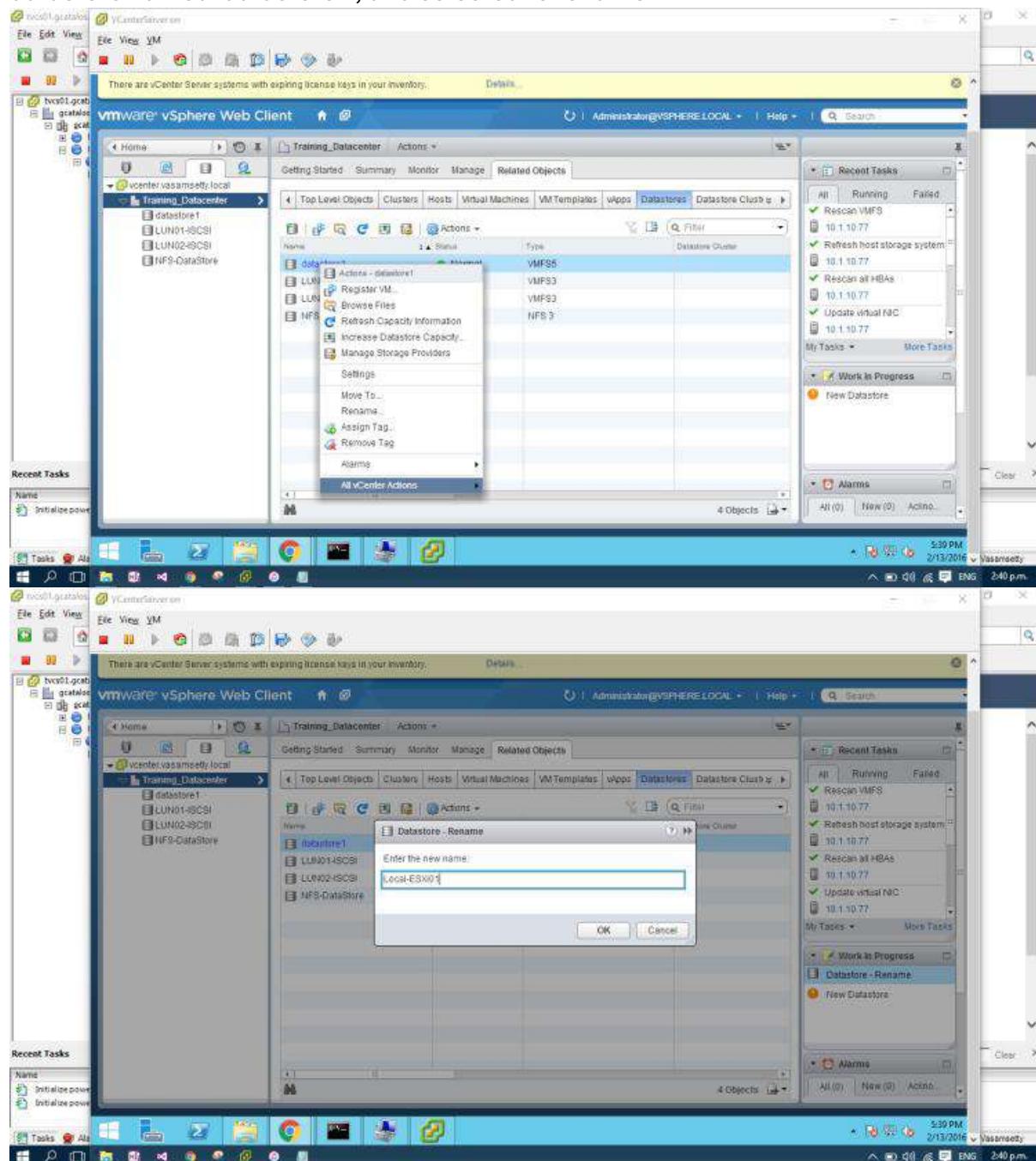
# Practical No 6

## Managing VMWARE Vsphere VMFS

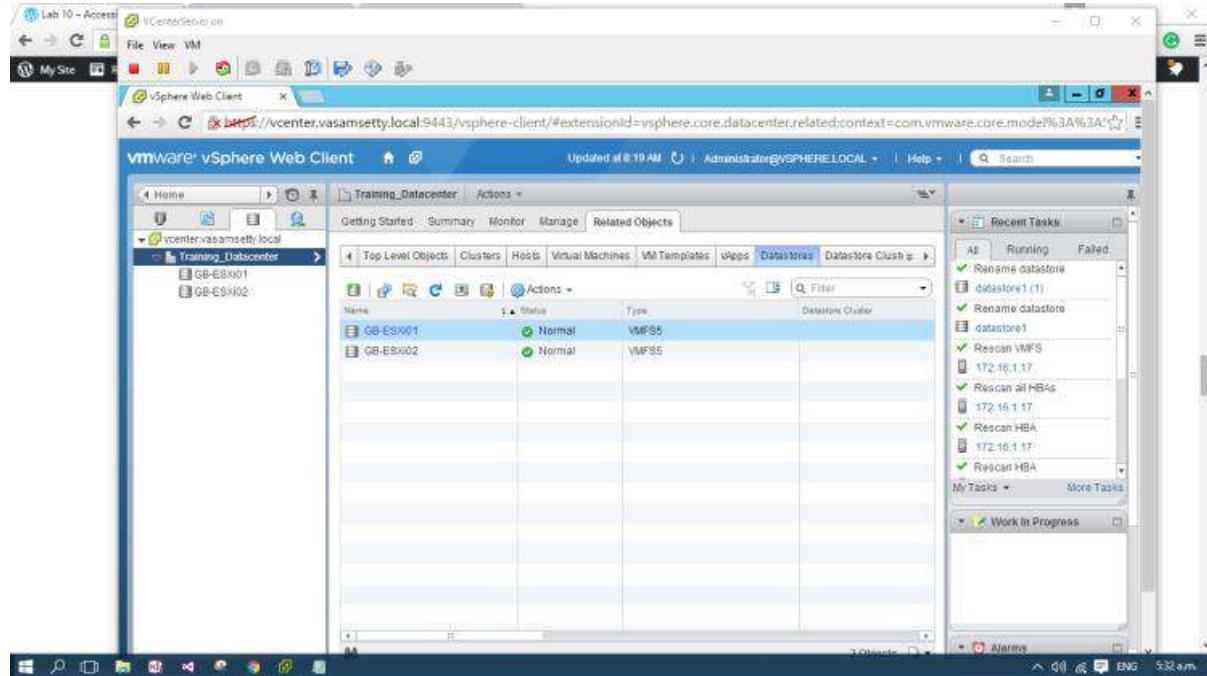
- Create a VMFS Datastore
- Expand a VMFS Datastore to Consume Unused Space on a LUN
- Remove a VMFS Datastore
- Extend a VMFS Datastore
- create a second shared VMFS datastore using iSCSI in VMware

### Changing the Name of a VMFS Datastore

Once you added your host devices to your vCenter you can able to see the datastore's list in the storage section. I navigated to 'vCenter -> Datastores', right-clicked on the datastore named 'datastore1', and selected to rename it.

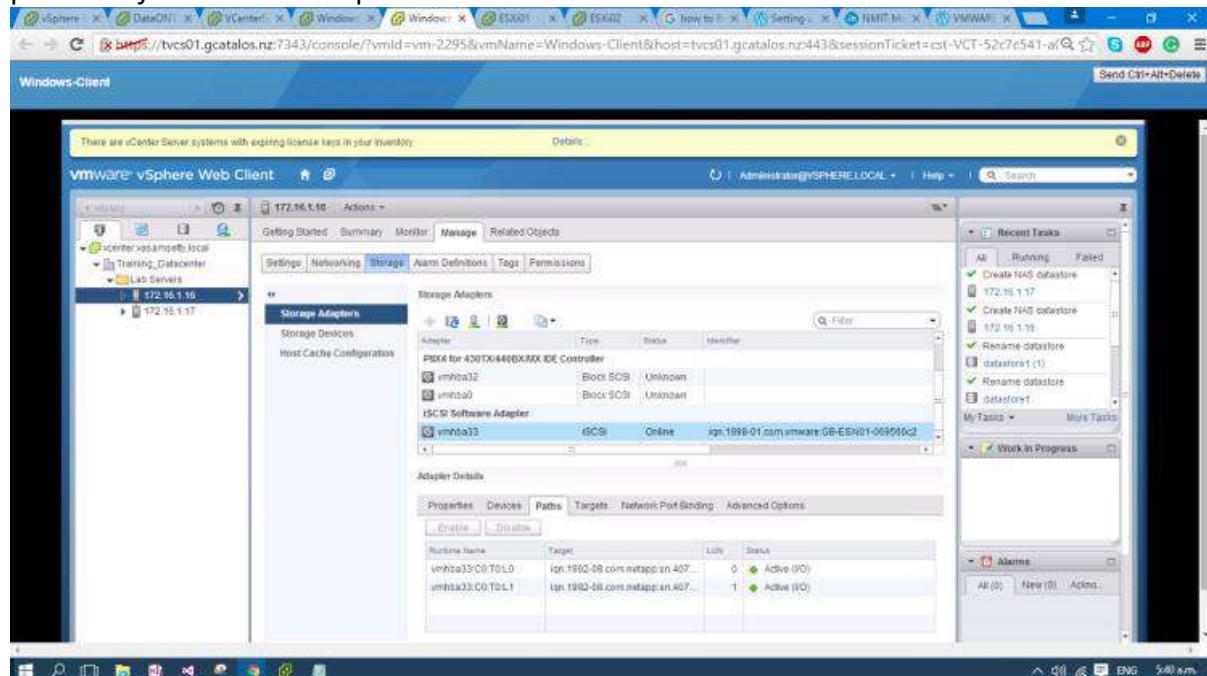


Once i renamed my two hosts datastore's i rechecked whether they have been changed or not. I found they were renamed successfully.



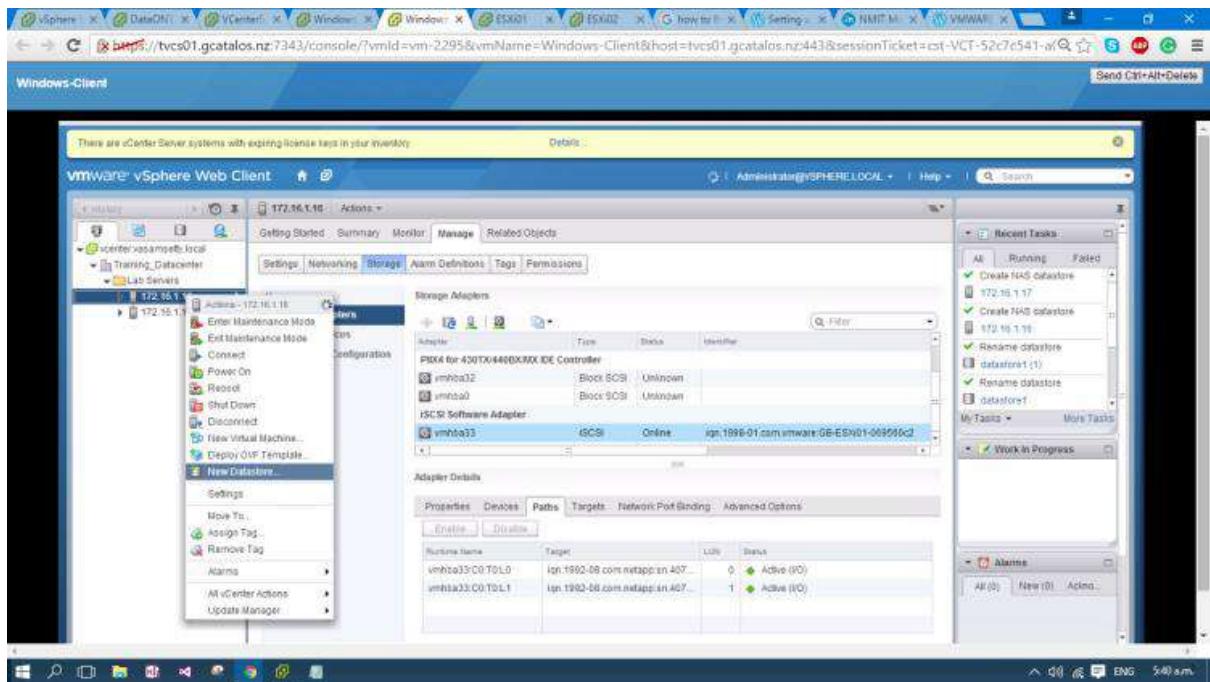
### Reviewing the Shared Storage Configuration

I carried out this task by navigating to 'vCenter -> Hosts and Clusters -> Lab Servers -> 172.16.61.44 -> Manage -> Storage', clicking on the iSCSI adapter, and clicking on the 'Paths' tab. From here I could see information about the LUNs that I had mapped previously under the 'Adapter Details' section.

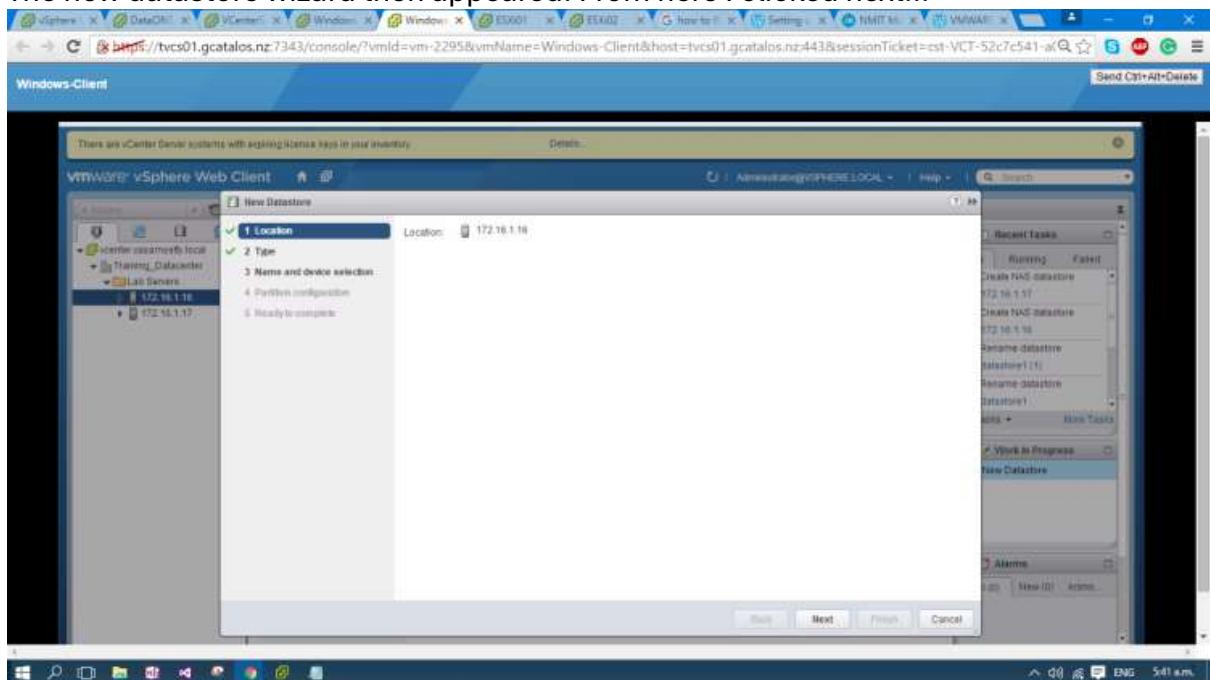


### Creating a VMFS Datastore

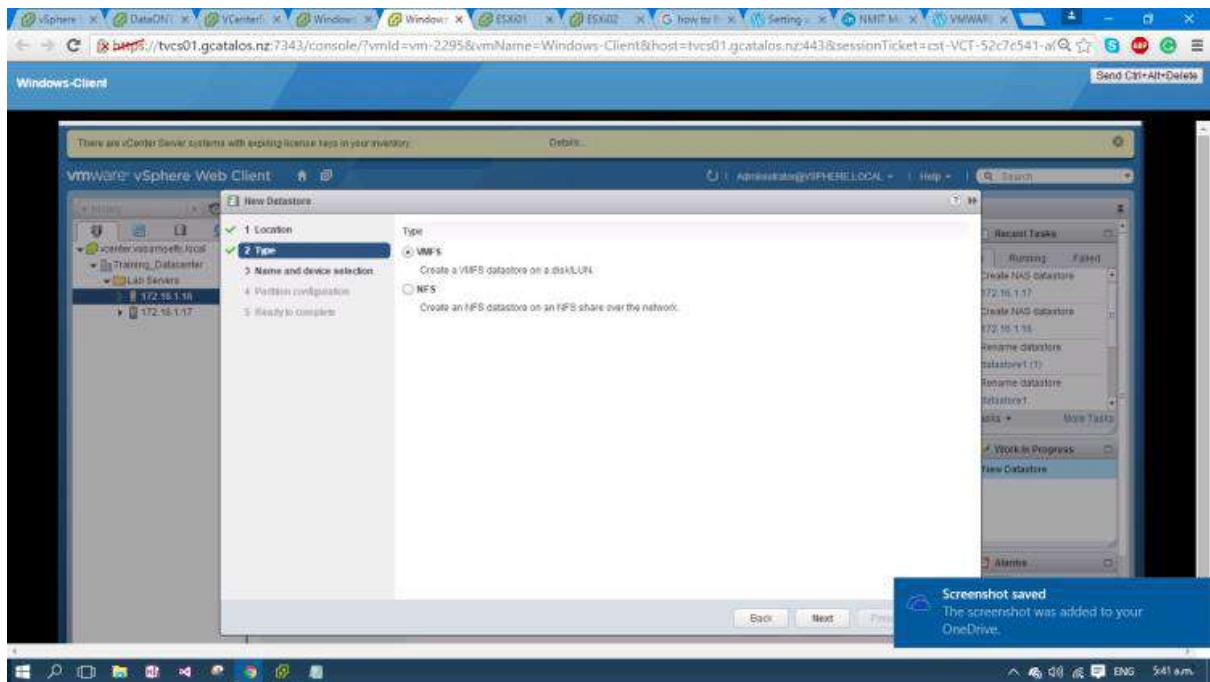
To create a new VMFS first I navigated to 'Home -> vCenter -> Hosts and Clusters -> Lab Servers and right clicked on my ESXi host and selected on New Datastore



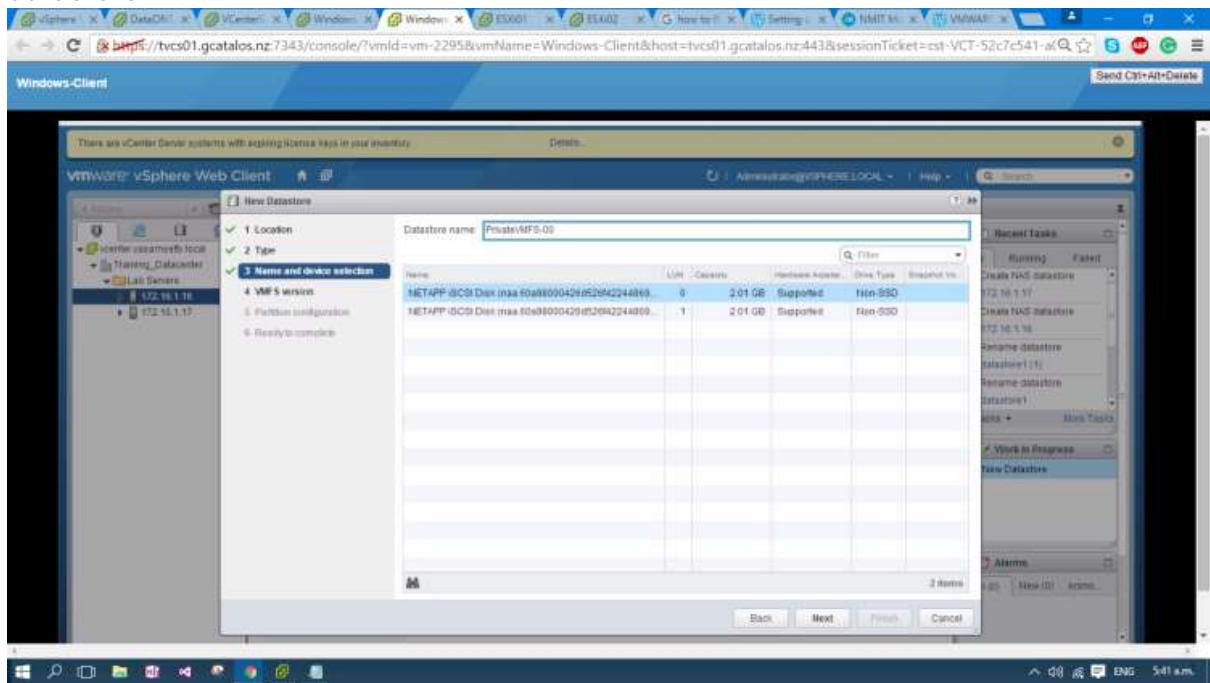
The new datastore wizard then appeared. From here I clicked next...



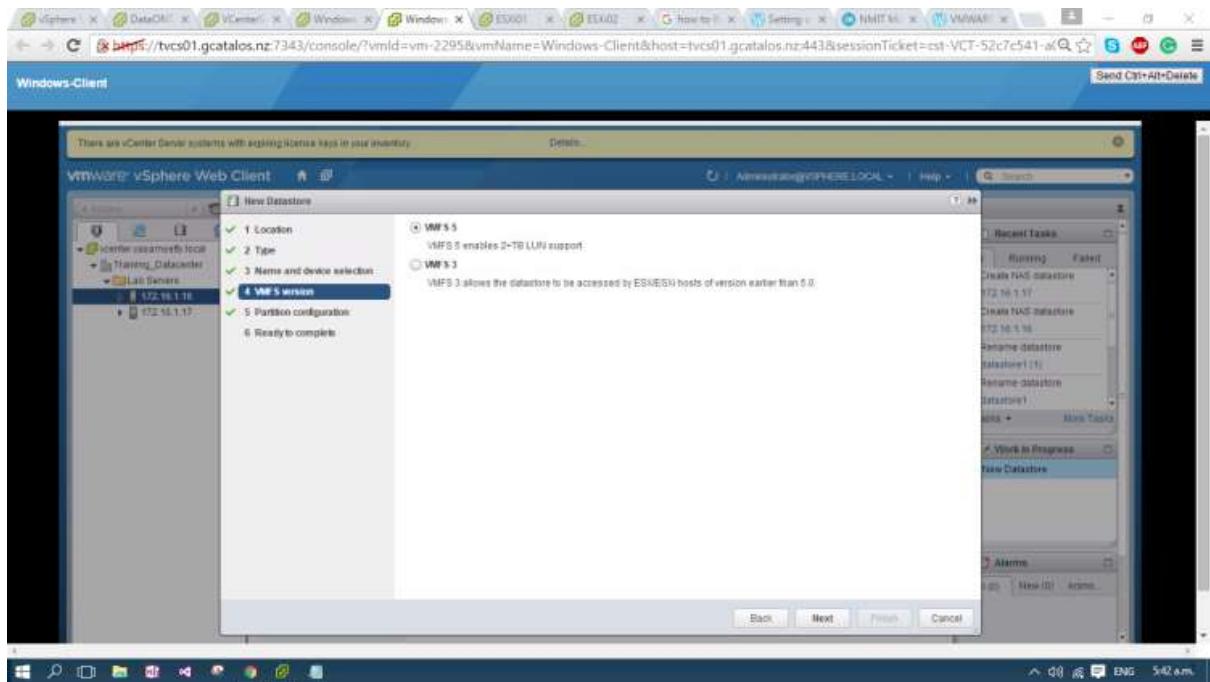
selected 'VMFS' as the type for the datastore...



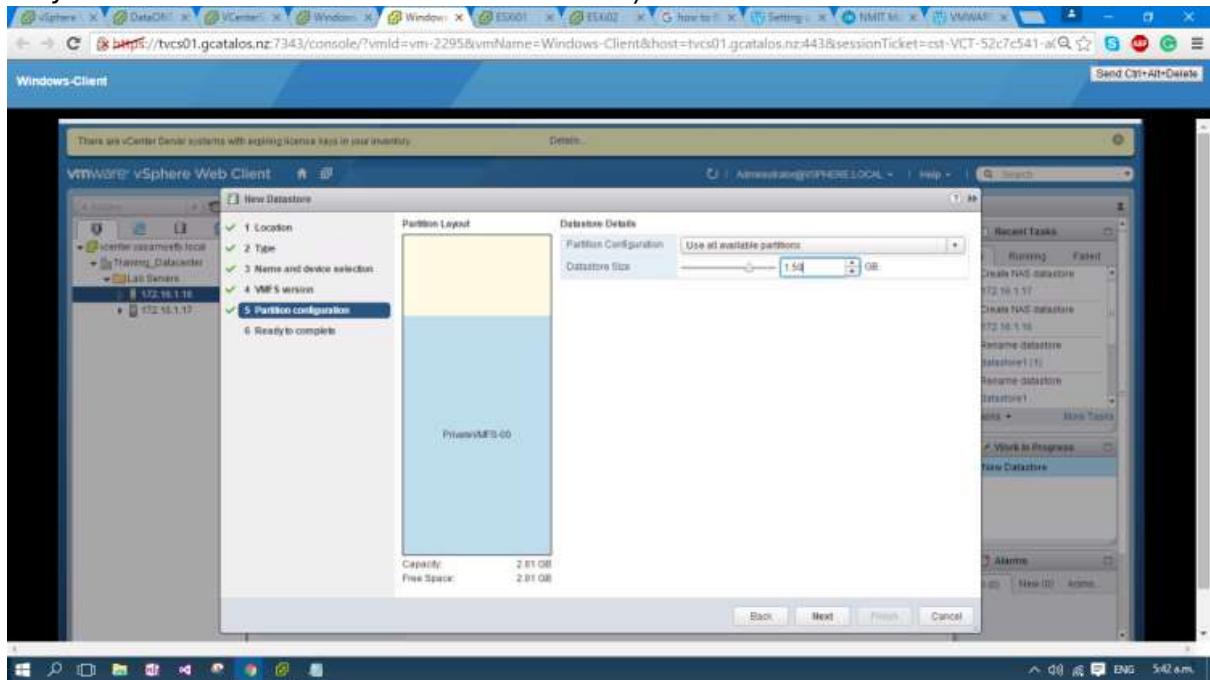
entered 'PrivateVMFS-00' for the datastore name and selected LUN0 as the LUN for the datastore



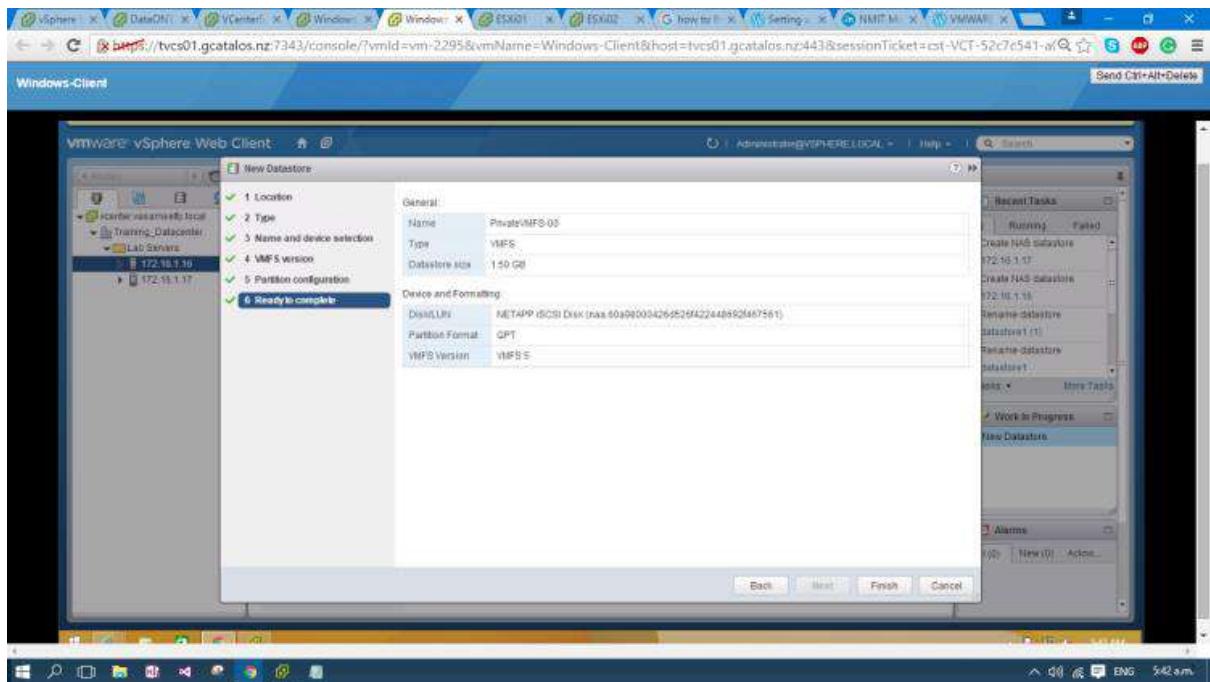
chose the default of 'VMFS 5' for the VMFS version



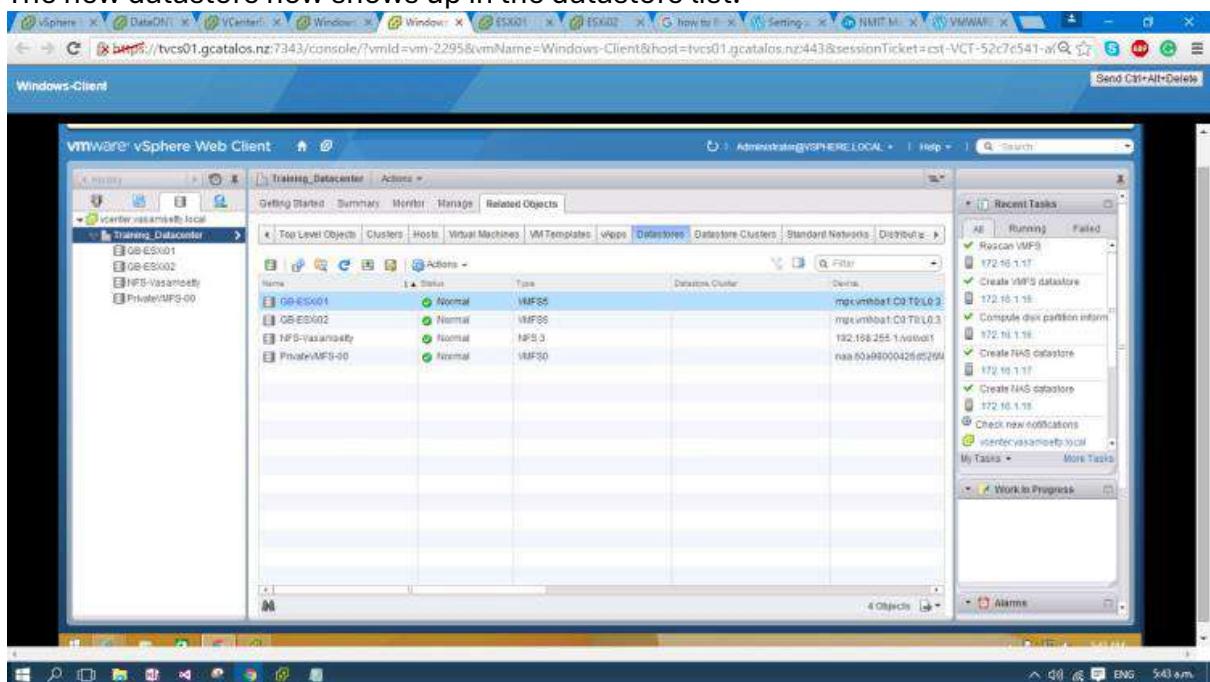
reduced the datastore size by 500MB (Note: It would not allow me to reduce the datastore size by 1GB as the lab manual suggested, probably because the LUN in use is only 2GB and there is some sort of restriction.)



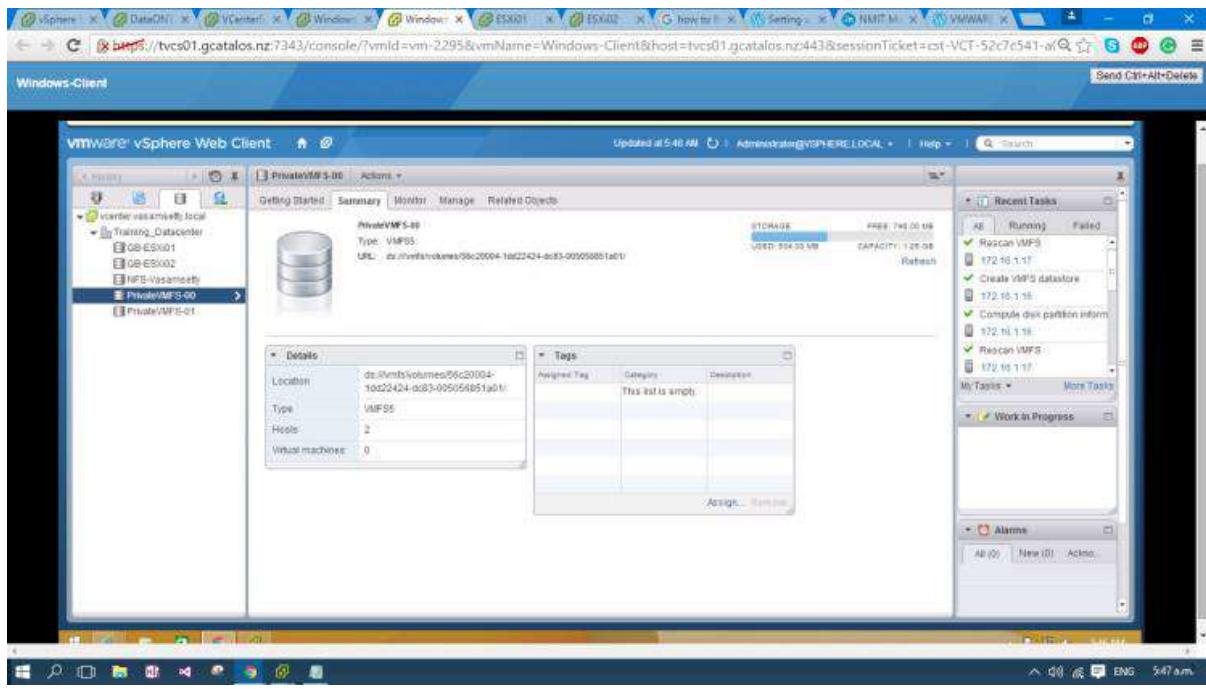
and finally, reviewed the settings and created the datastore.



The new datastore now shows up in the datastore list.

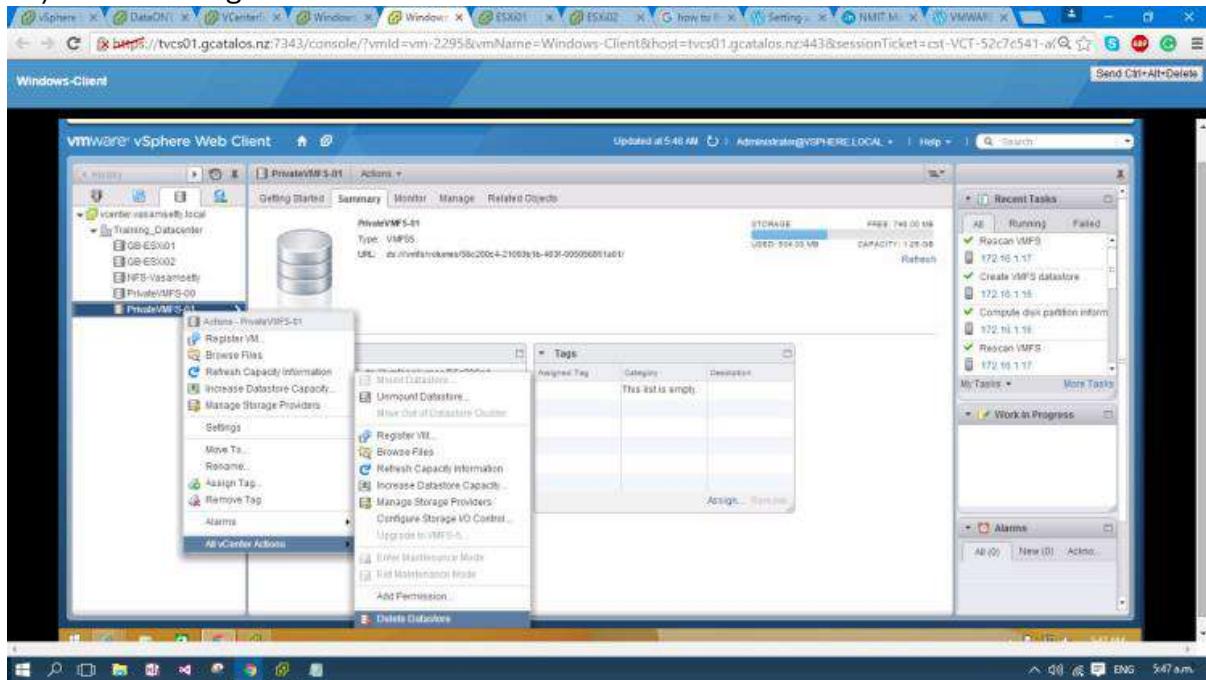


I then repeated these steps to add another datastore using my other LUN. I changed only the name of the datastore, the LUN to use, and I did not make any changes to the partition.

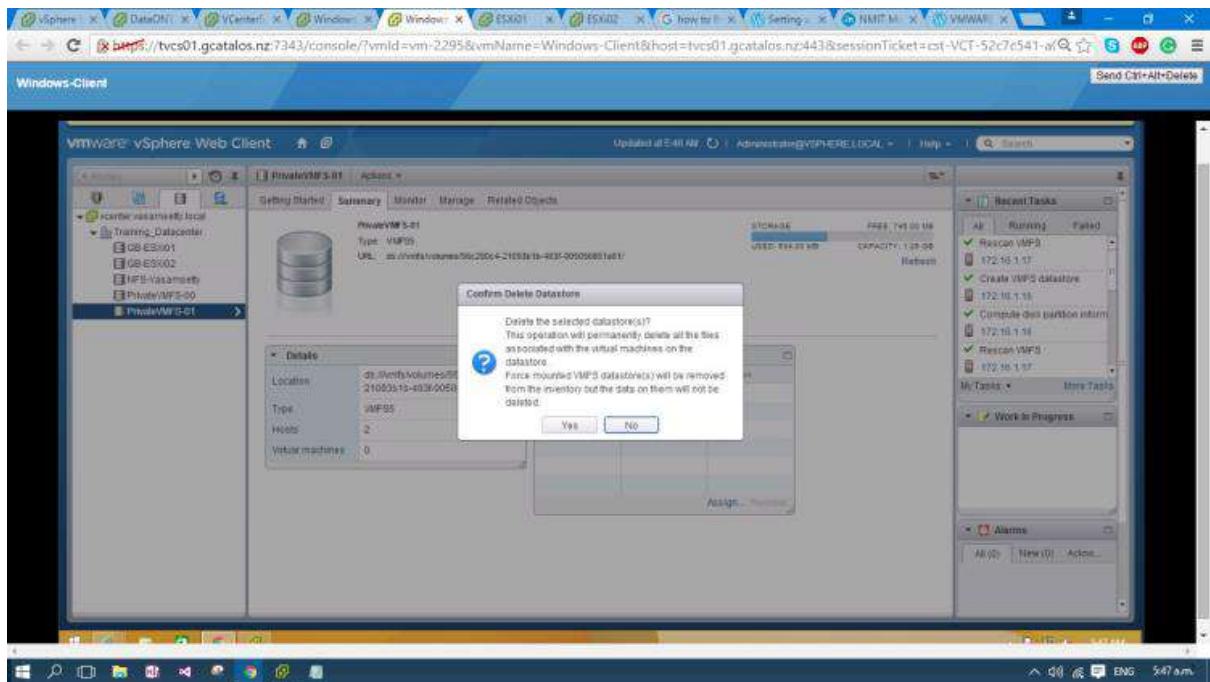


## Removing a VMFS Datastore

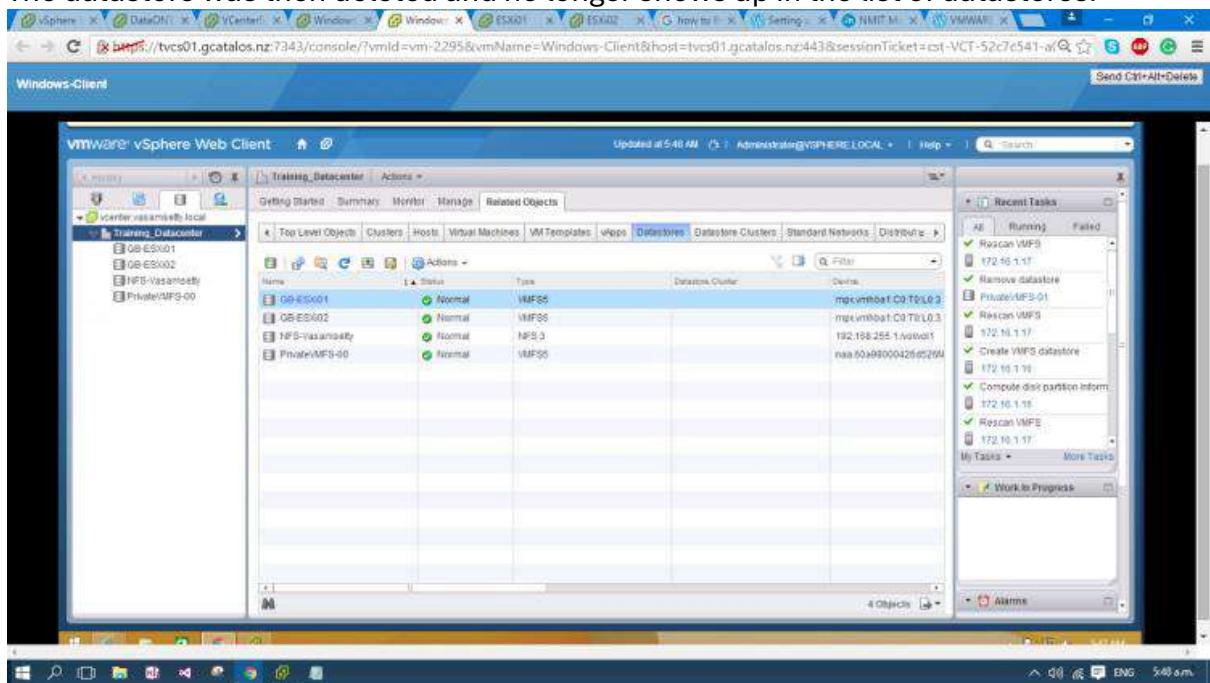
I started this task by right-clicking on the second datastore that I created (PrivateVMFS-01) and selecting 'All vCenter Actions -> Delete Datastore'.



This opened a prompt to confirm the deletion on the datastore. I clicked 'Yes'.



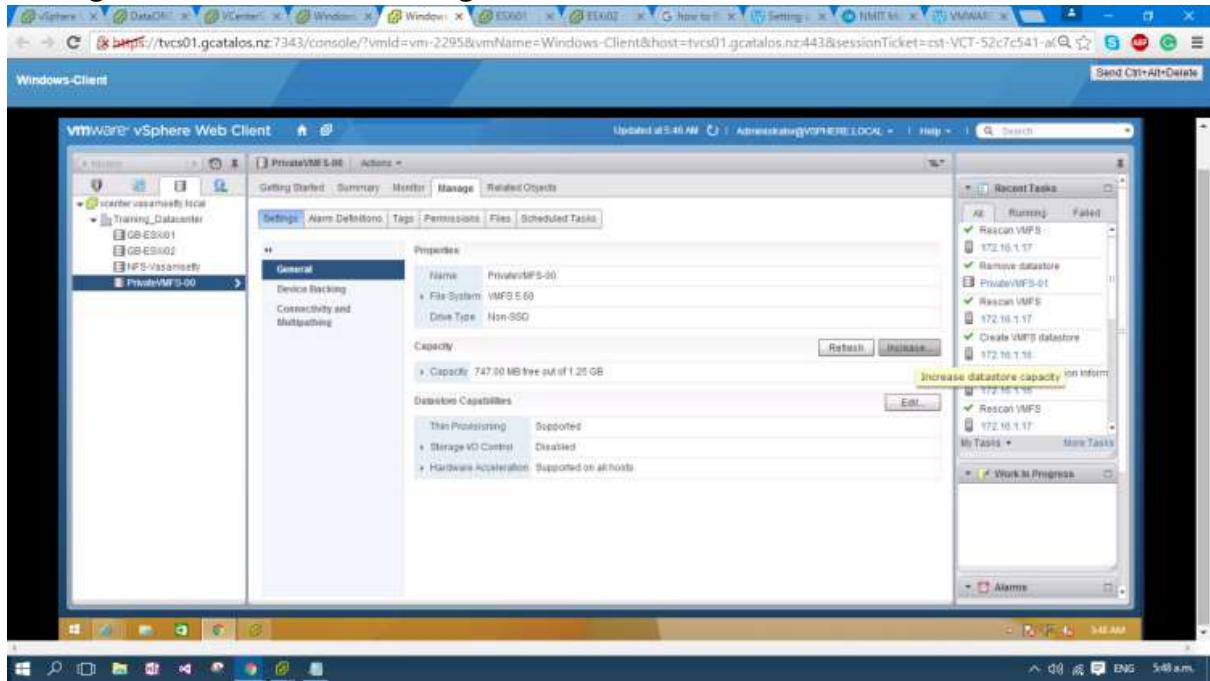
The datastore was then deleted and no longer shows up in the list of datastores.



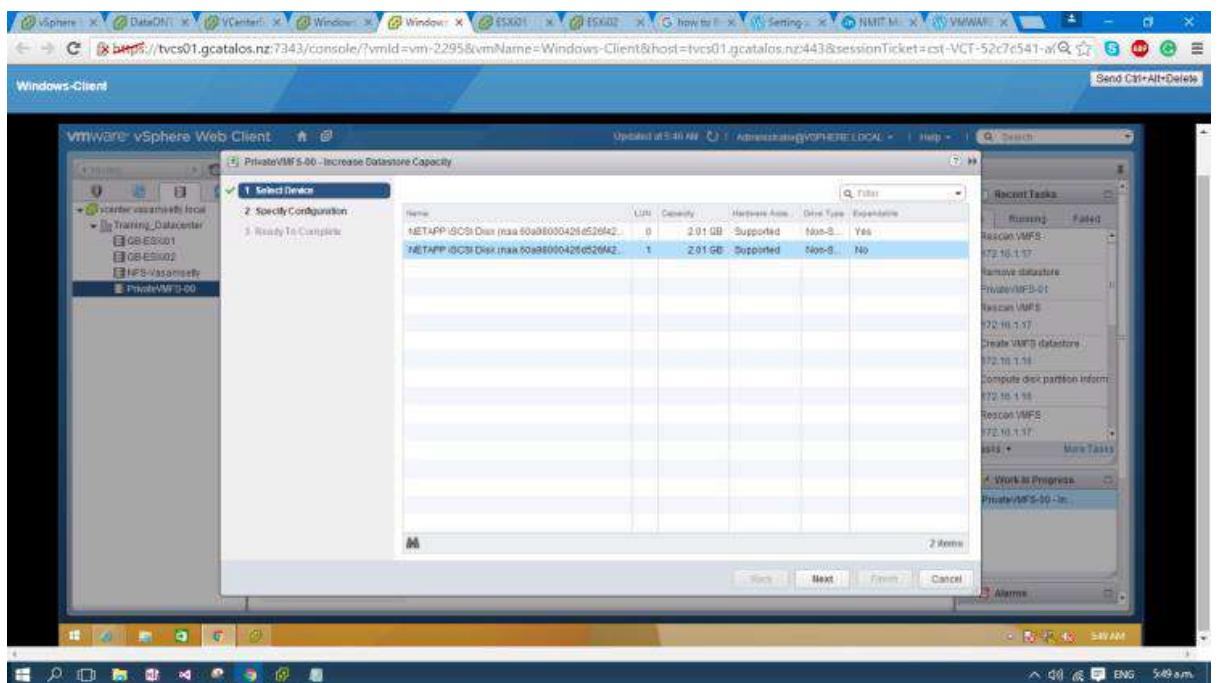
## Extending a VMFS Datastore

I began this task by selecting my PrivateVMFS-00 datastore, navigating to 'Manage ->

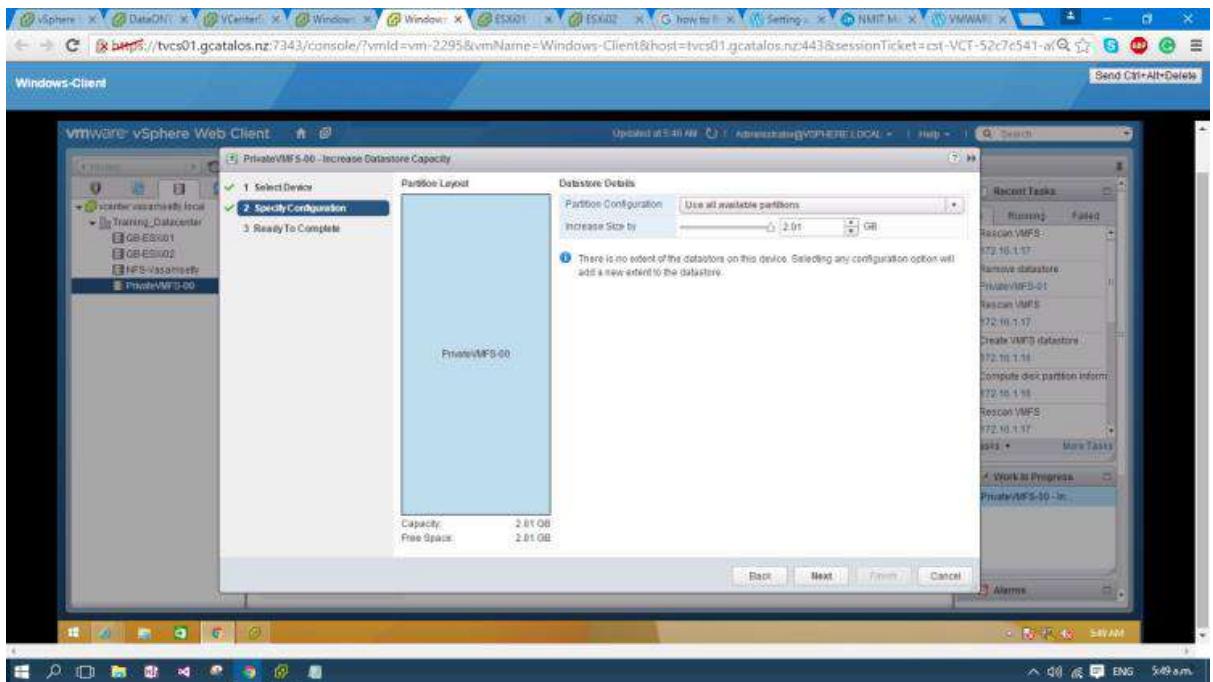
Settings -> General', and selecting the 'Increase...' button.



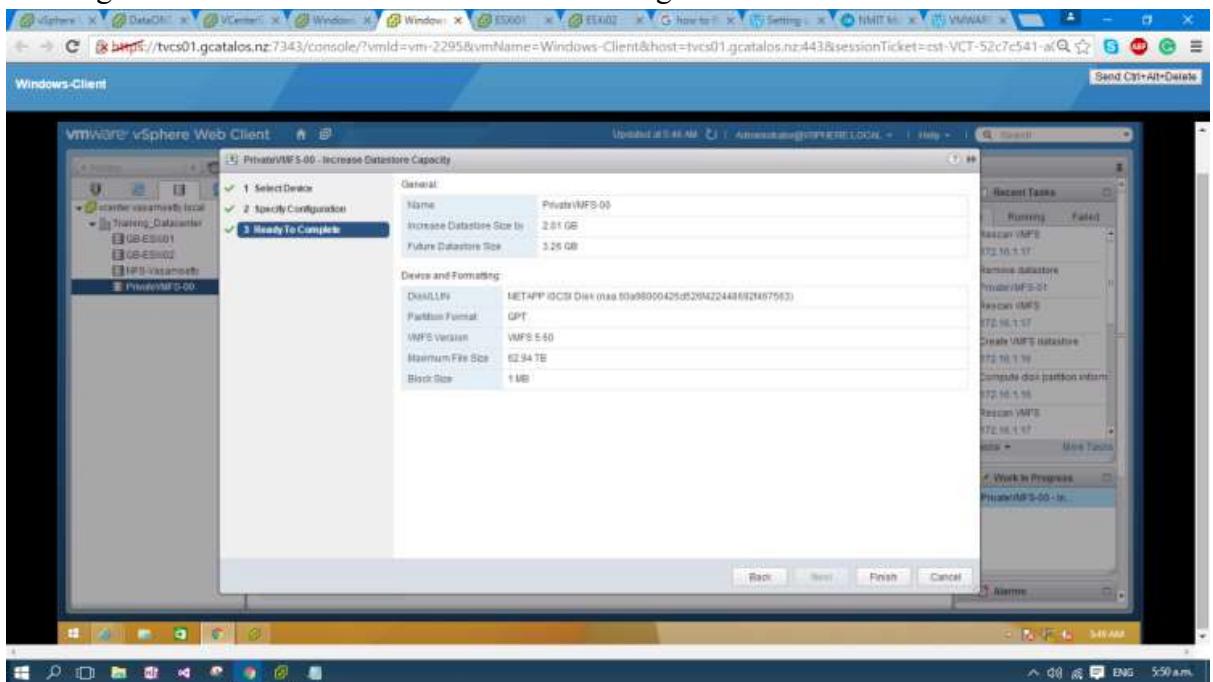
From here, I selected the LUN that is now available due to the other datastore having been deleted.



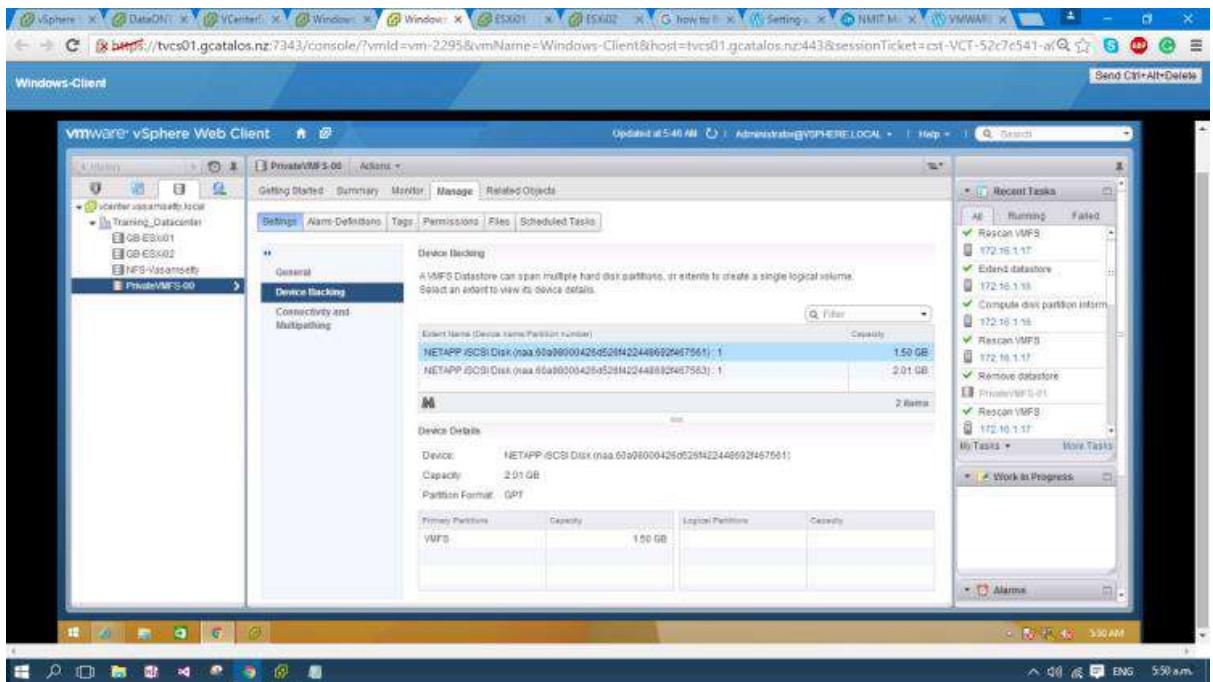
I then selected 'Use all available partitions' from the drop down list. This should now take advantage of the spare LUN.



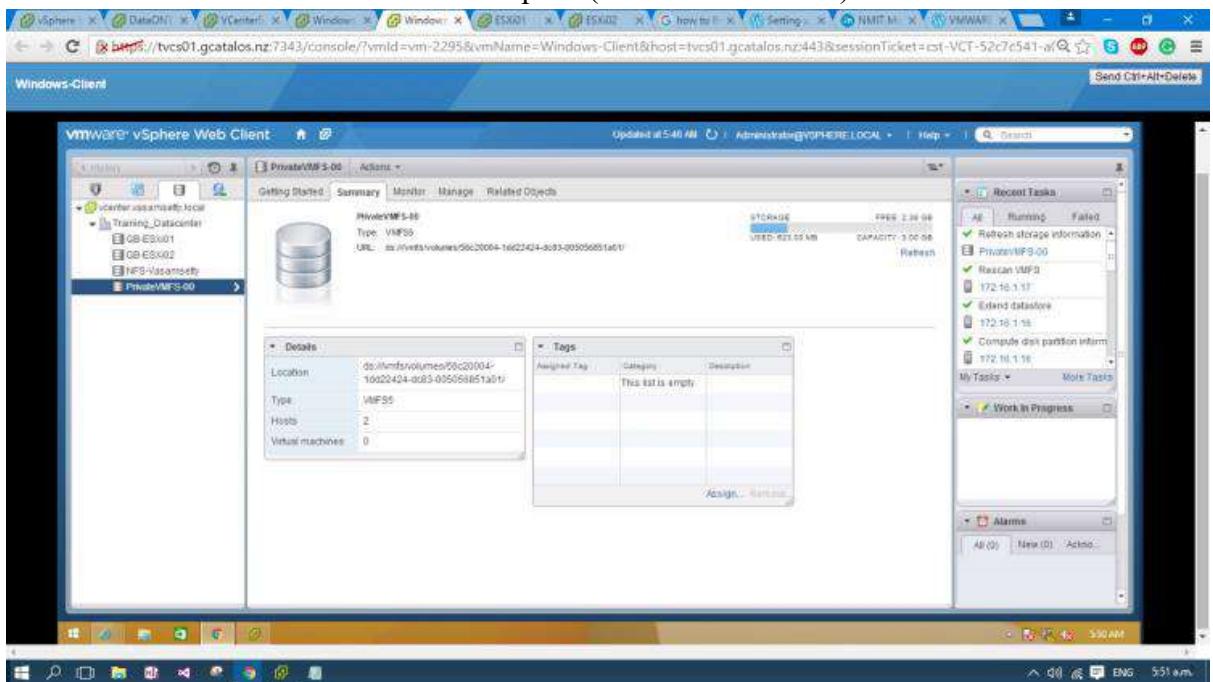
After finishing the review and completing the resizing, I navigated to the ‘Device Backing’ tab to see if both of the extents were being used.



And sure enough, they both showed up.

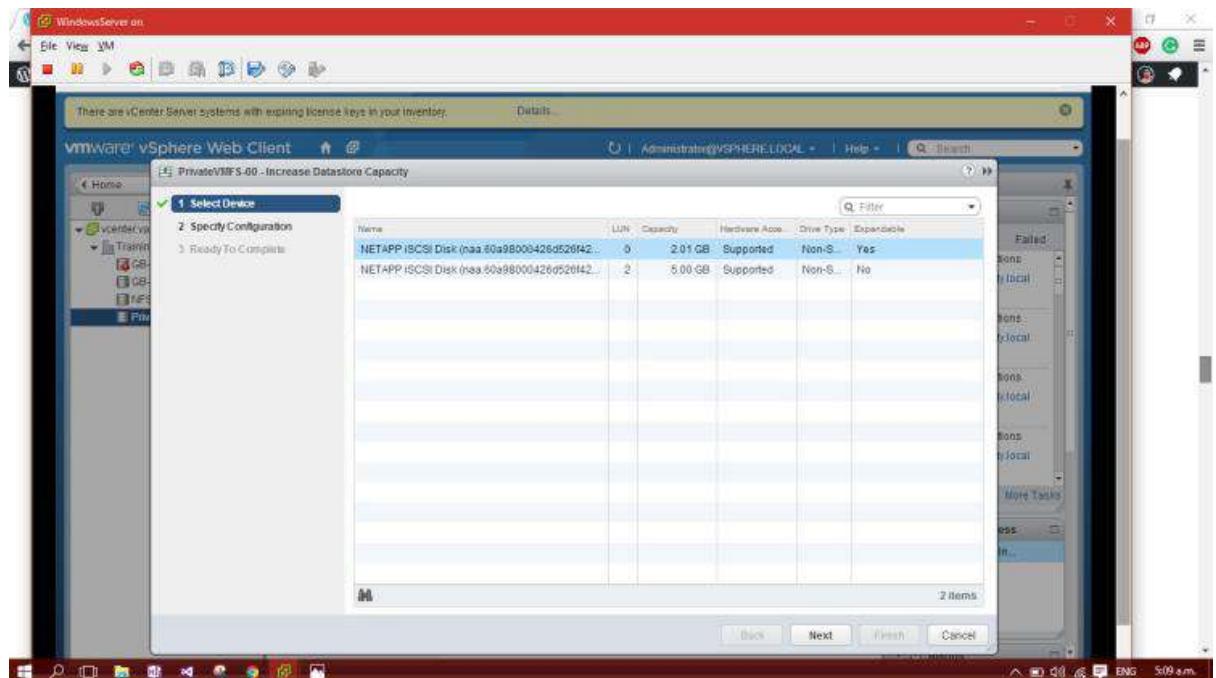


I then checked out the information under the ‘Summary’ tab. The datastore now uses both of the LUNs for a total of 4GB of space (minus reservations).

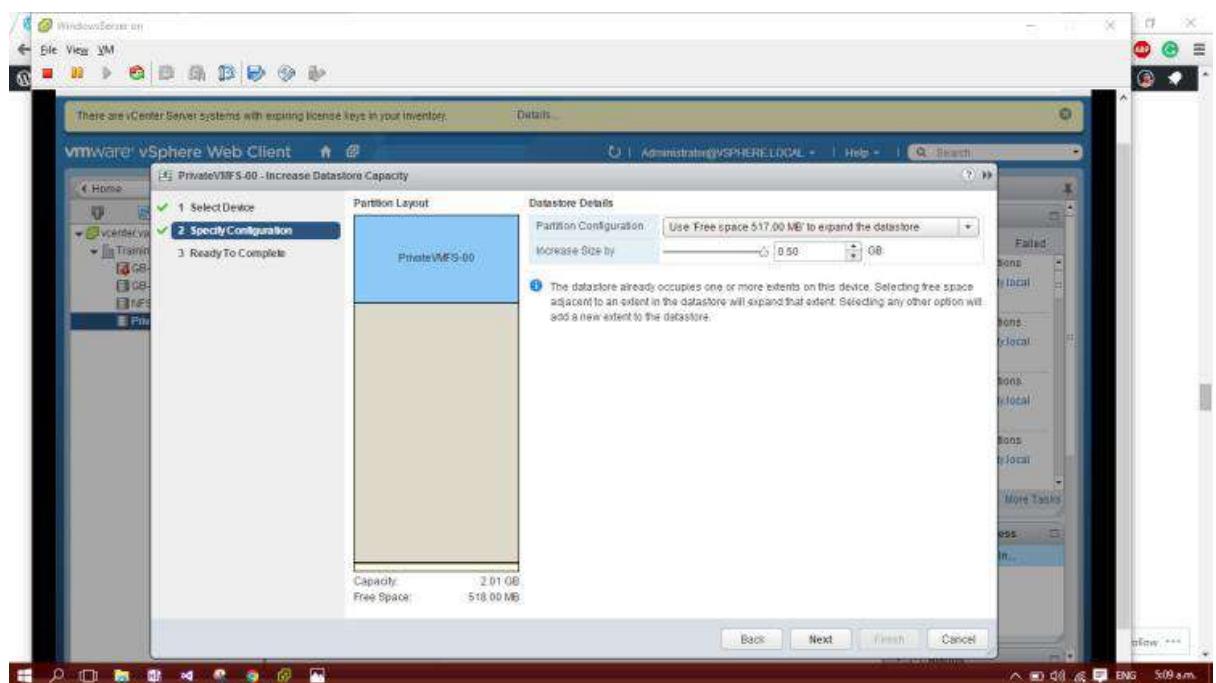


### Expanding a VMFS Datastore to Consume Unused Space on a LUN

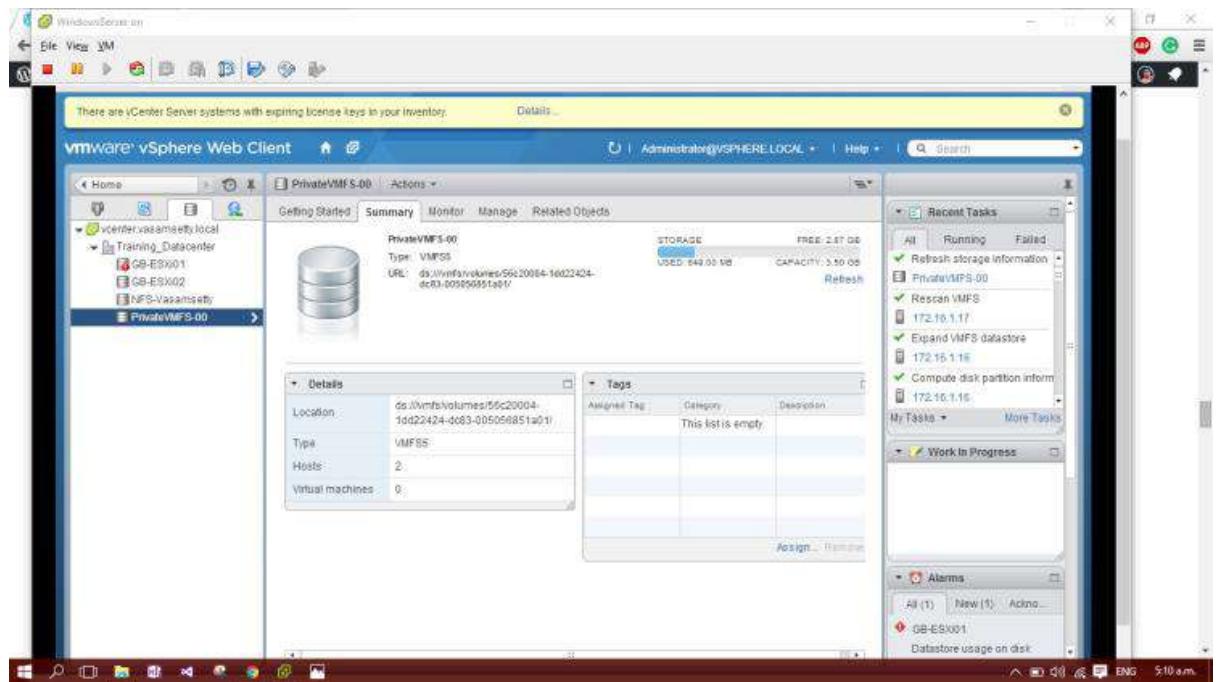
I began this task by selecting my PrivateVMFS-00 datastore, navigating to ‘Manage -> Settings -> General’, and selecting the ‘Increase…’ button. From here, I selected the LUN that this datastore uses and clicked next.



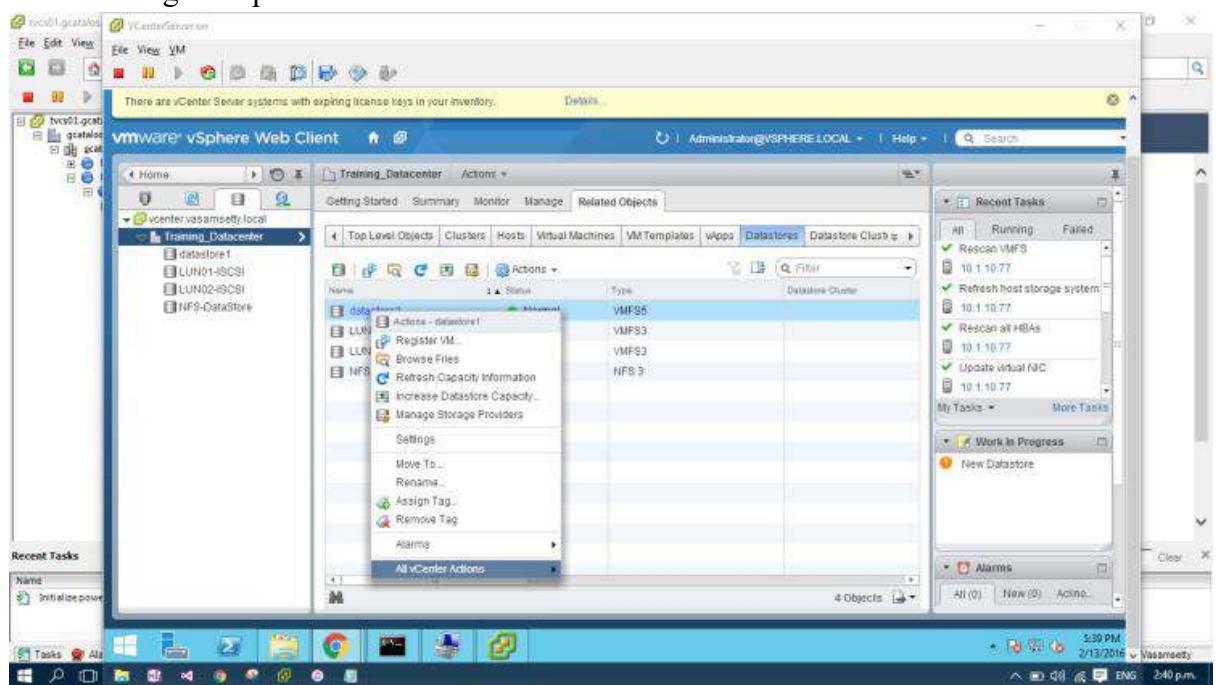
I then selected “Use ‘Free space 511.00 MB’ to expand the datastore” from the dropdown list and clicked next. This should now take advantage of the full 2GB available on the LUN.



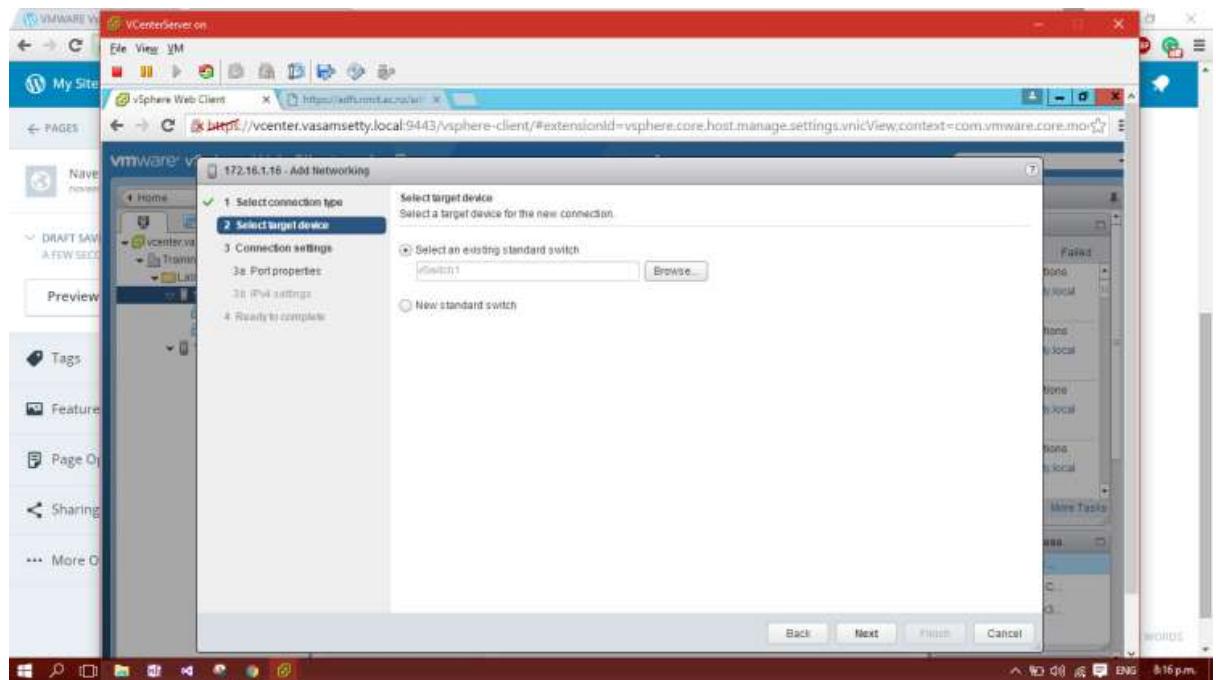
After finishing the review and completing the resizing, I checked out the information under the ‘Summary’ tab. The datastore now uses all 2GB available (minus a few MBs for the reservation).



Select Storage Adapters.



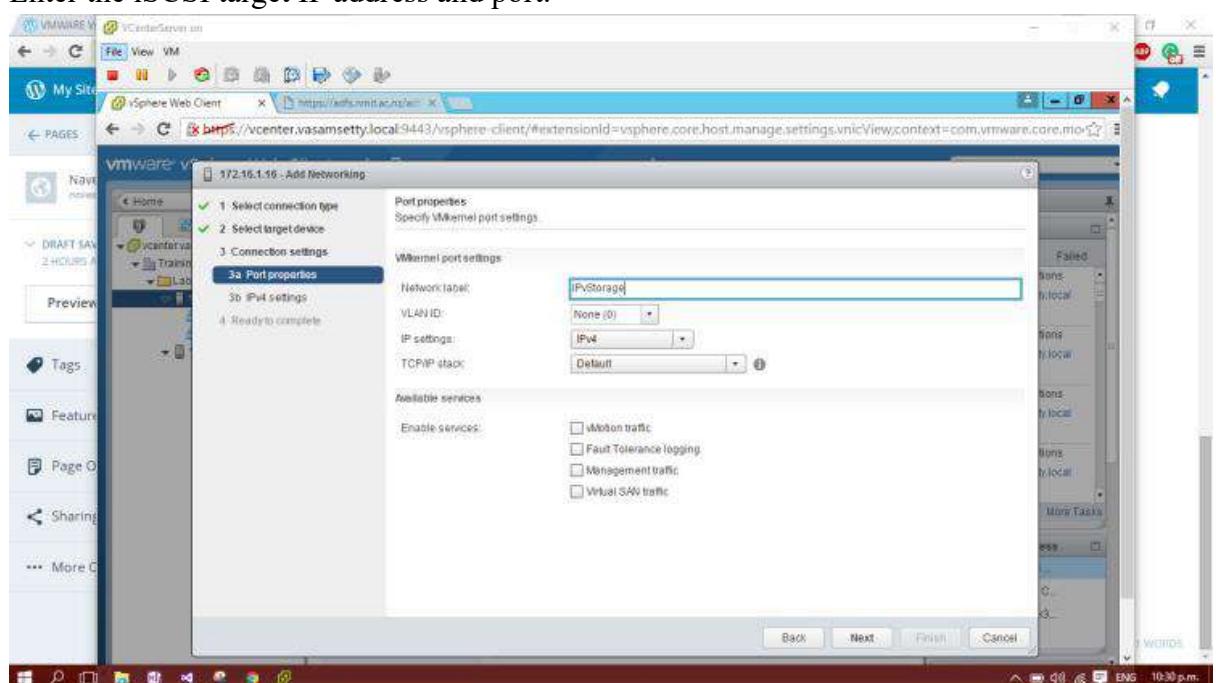
Click on the iSCSI Software Adapter.



Click on the Targets tab.

Click on the Add Target button.

Enter the iSCSI target IP address and port.



Click OK.

### Practical 7: Accessing NFS Storage

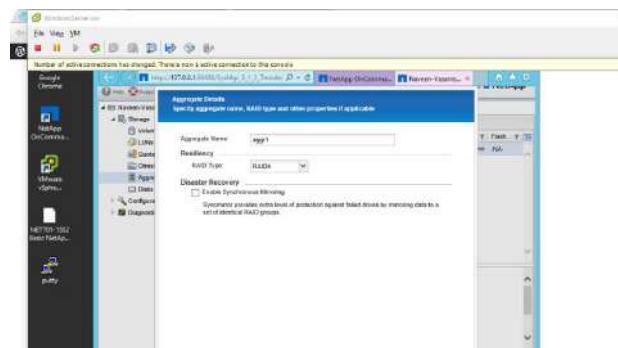
In this lab, I was required to do the following tasks:

Configure Access to NFS Datastores

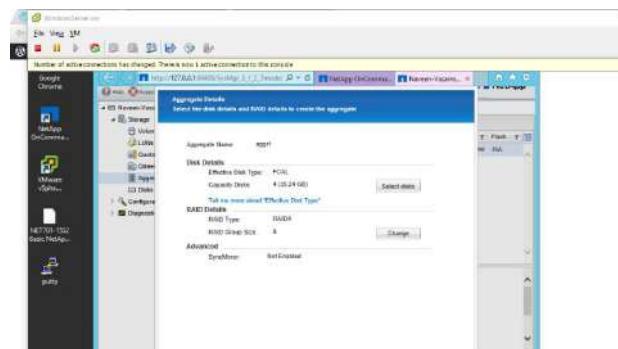
View NFS Storage Information

To create a new aggregate along with a new volume to be used as the NFS datastore. I began by enabling TLS on my vSIM as I plan to use the NetApp OnCommand System manager to create the aggregate and volume. My first task is to create an aggregate for this I then navigated to 'Storage -> Aggregates' and created a new aggregate. For the aggregate name I chose 'aggr1', for the RAID type I chose RAID-DP, and I left the enable synchronous mirroring tick box unchecked.

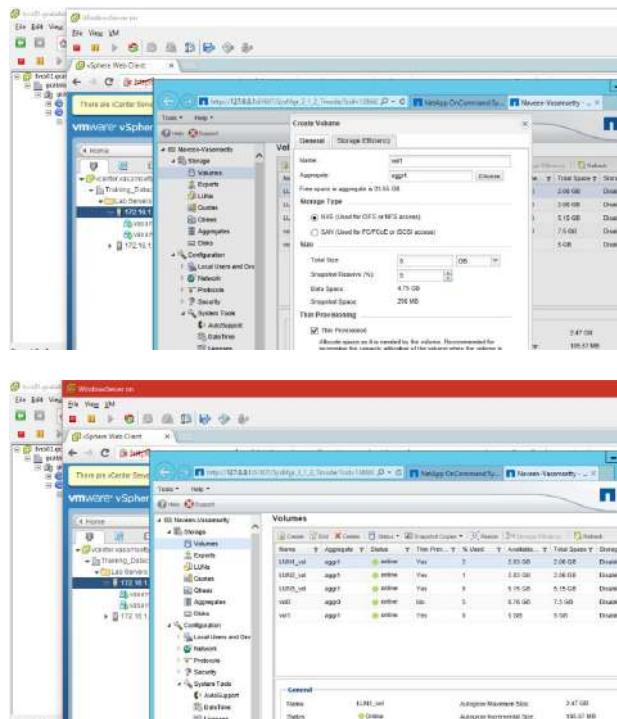
NOTE: Please make sure you added the NFS License and Started the NFS service on DATA ONTAP.



then needed to select the number of disks to assign to the aggregate. I choose 4 as my magic number because i got total 5 disks and I keep one disk un-partitioned for any fail overs.

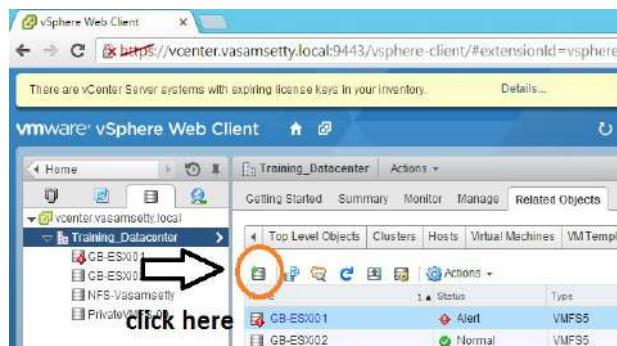


After I successfully create my aggr1, in the next phase I need a vol name vol1 to be created which acts as an NFS data storage. In order to do this, I navigated to 'Storage -> Volumes', clicked on create. This brought up the Create Volume wizard. For the name I entered 'vol1', for the aggregate I selected the newly made 'aggr1', for the storage type I selected 'NAS' (as this is for NFS access), for the size a specified 5GB, and finally, I selected to make the volume thin provisioned.

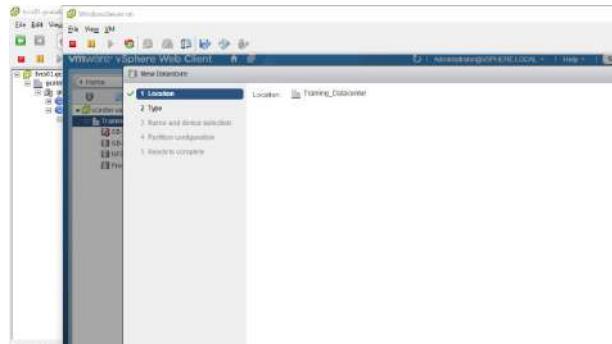


### Configure Access to NFS Datastores

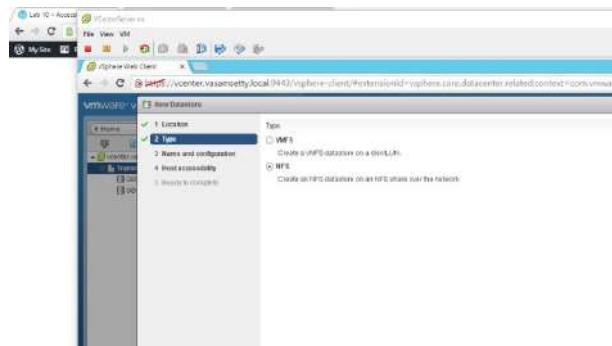
Now its time to configure NFS Datastores. Once I was in, I navigated to 'vCenter -> Storage -> Related Objects' and clicked on the 'Create a new datastore' button.



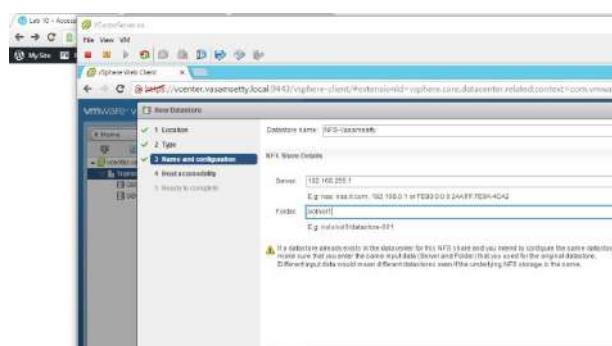
this will set me to a wizard of configuring the NFS-datastore. In my first window I selected my datacenter that are available to my network.



For the type, I chose NFS.

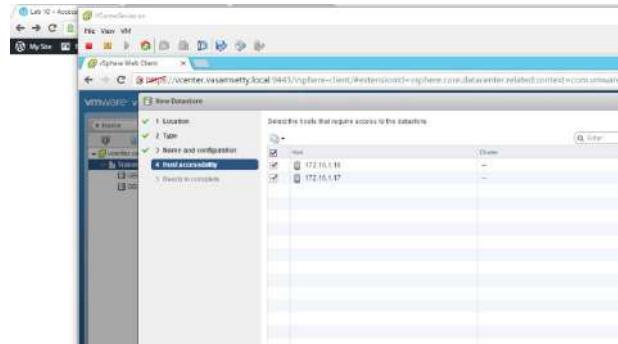


For the datastore name, I followed the lab manual naming scheme of 'NFS-<your-name>' and entered 'NFS-Vasamsetty'. For the server, I put in the IP address of the NetApp vSIM on the 192.168.255.1 network. Finally, for the folder path, I entered the path of '/vol/vol1', pointing to the volume that I created earlier in the post. I chose not to mount the NFS as read-only so that I can actually make changes to it.



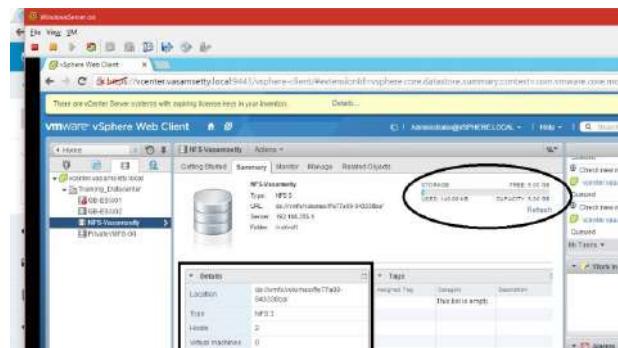
As far as accessibility to the datastore goes, I allowed both of my ESXi hosts to access the datastore.

After making sure the values in the review were correct, I finalised the creation.



#### View NFS Storage Information

Once my storage was successfully created. I can see the storage information and its summary right over on selecting NFS-VASAMSETTY



#### Lab 12 : Using Templates and Clones

In this lab, I was required to do the following tasks:

Copy Sysprep Files to vCenter Server Appliance

Create a Template

Create Customisation Specifications

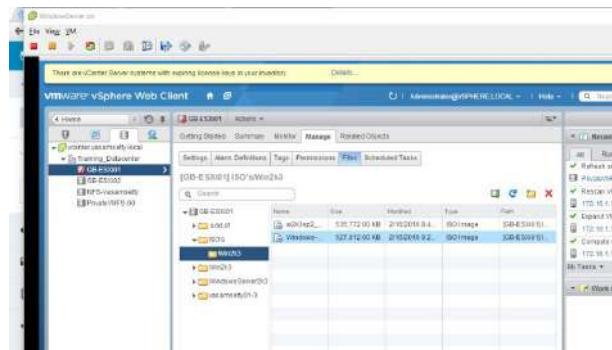
Deploy a Virtual Machine from a Template

Clone a Virtual Machine that is Powered On

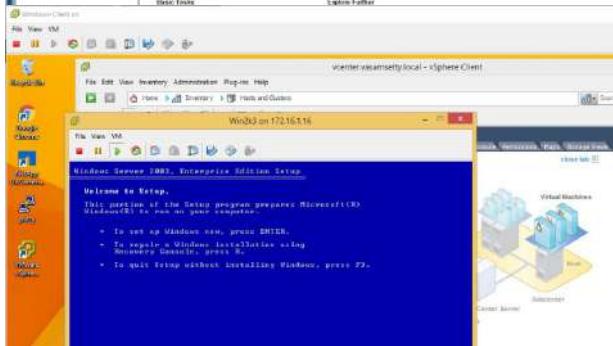
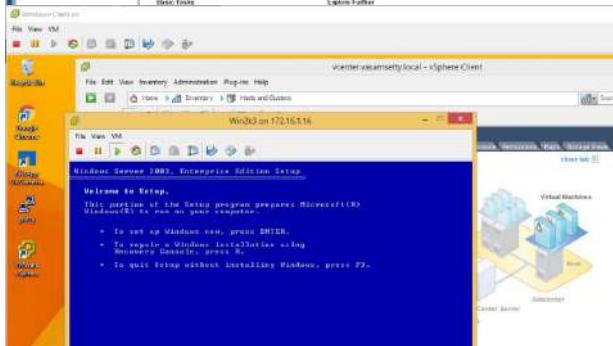
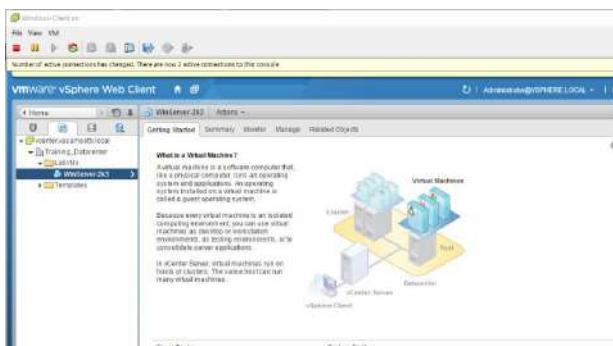
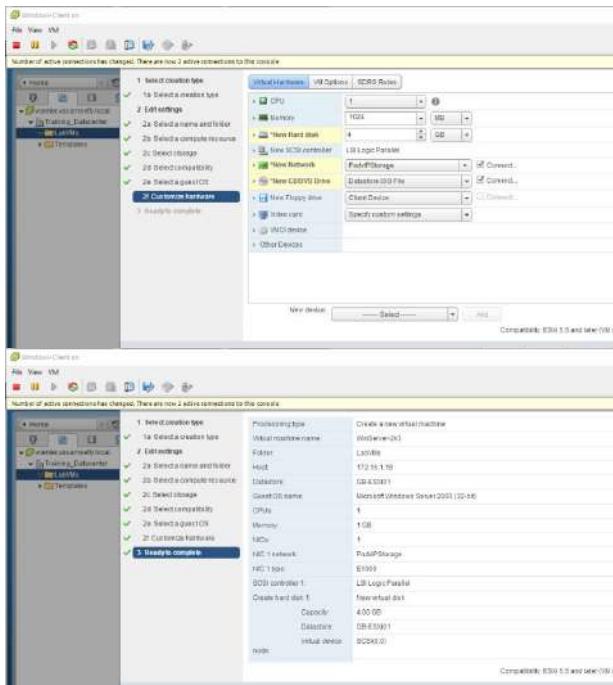
NOTE: Before uploading any type of data to your datastore we need to make sure that we had installed VMware Client Integration Plugin in your browser.

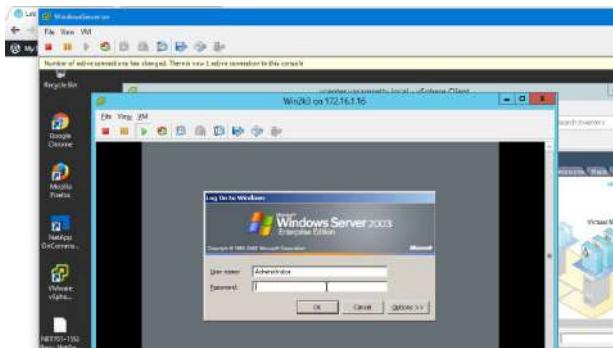
Pre-requirements: Make sure you uploaded your MicroSoft windows 2003 server ISO in your datastore. I tried very hard to do so, but after the perfect upload i feel really excited to do my further labs that because am gonna create a new network inside a builtin network its like a “double virtualisation” its really fantastic.

As you can see I just uploaded my Microsoft 2003 server into my datastore.



Now am ready to install a VM on my host. As discussed in our previous lab ” how to install the guest operating system” i installed my MS 2k3 and powered it on.

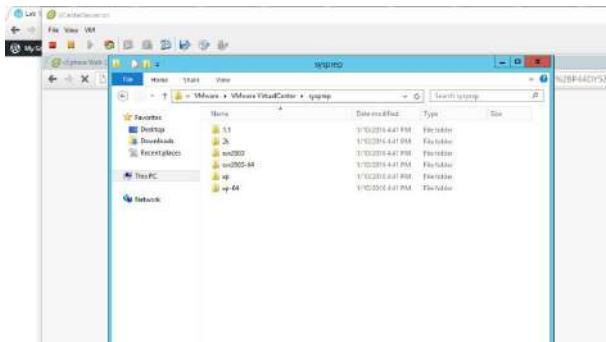




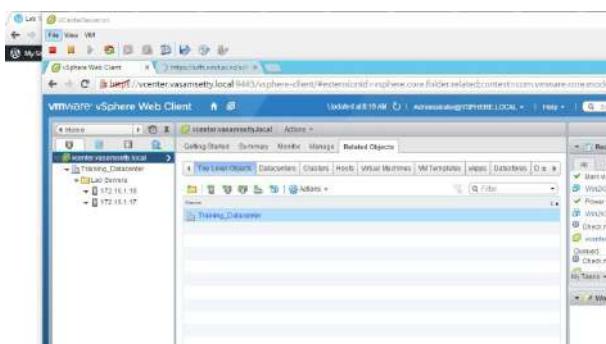
that's a successful milestone. Now i had a active windows server 2003 running VM and am gonna make a template of it and clone it.

#### Copying Sysprep Files to vCenter Server Appliance

As we don't have specific access to confined folders and I had not yet set up the Vcenter Server Appliances I can't copy the files right now. At this stage i can able to find the Sysprep folder and tried to learn what are files in it.



then navigated to 'vCenter -> VM Templates' and the new template showed up in the list.

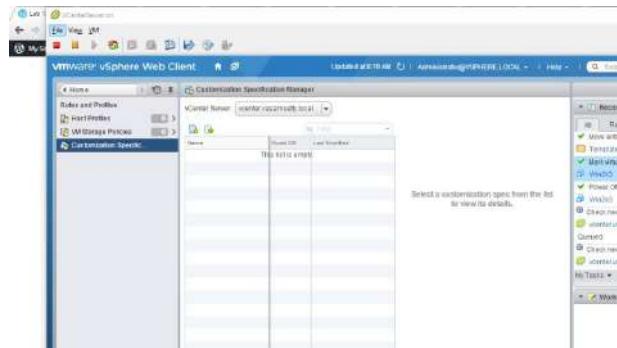


I just moved my created VM into VM's folder that i had created in my previous lab.

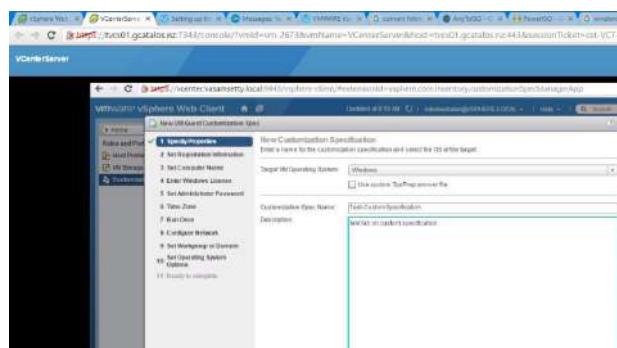
got a template creation wizard and finished it and finally, I right-clicked the template and renamed it to 'Win2k3' as per the VMware lab instructions.

### Creating Customisation Specifications

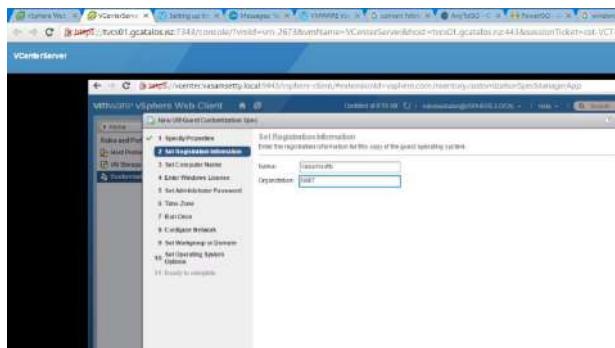
For this i navigated to the 'Customization Specification Manager' tab. I then clicked on the icon for creating a new specification.



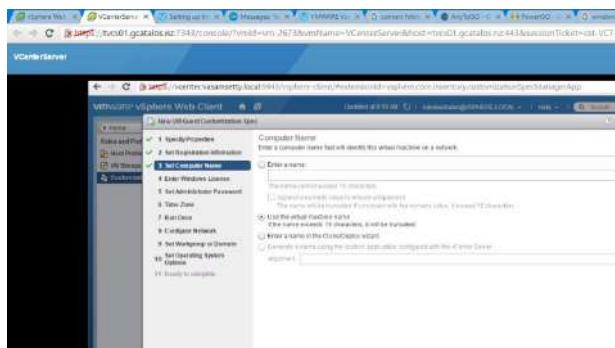
This opened the 'New Customization Specification' wizard. For the Target VM OS type, I selected Windows. For the name, I entered 'Test-CustomSpecification' as per the lab book instructions. I also added in a description.



For the name and organisation I put in some appropriate values



For setting the computer name, I selected the 'Use the virtual machine name' radio button and continued.

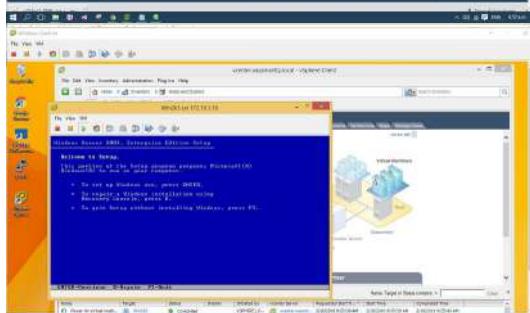
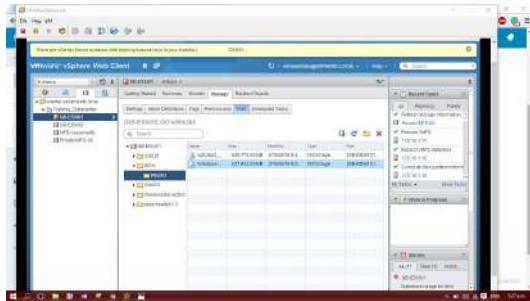


**Practical 8 : Using Template and Clones**

- Create Customisation Specifications
- Deploy a Virtual Machine from a Template
- Clone a Virtual Machine that is Powered On

**Pre-requirements:** Make sure you uploaded your Microsoft windows 2003 server ISO in your datastore. I tried very hard to do so, but after the perfect upload i feel really excited to do my further labs that because am gonna create a new network inside a built in network its like a “double virtualisation” its really fantastic.

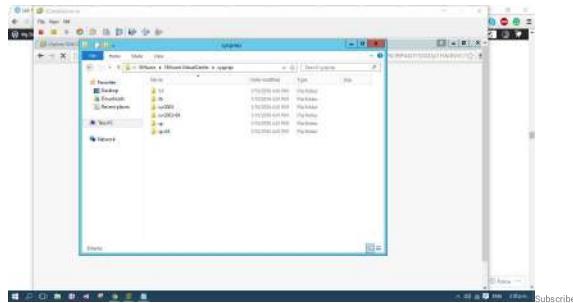
As you can see I just uploaded my Microsoft 2003 server into my datastore.



that's a successful milestone. Now i had a active windows server 2003 running VM and am gonna make a template of it and clone it.

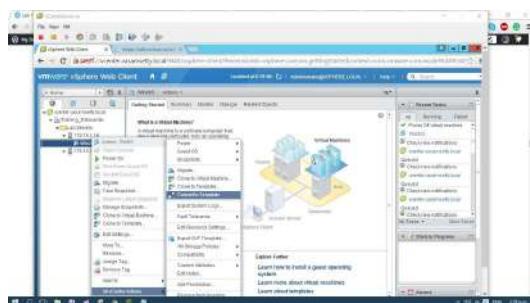
#### Copying Sysprep Files to vCenter Server Appliance

As we don't have specific access to configured folders and I had not yet set up the vCenter Server Appliances I can't copy the files right now. At this stage I can able to find the Sysprep folder and tried to learn what are files in it.

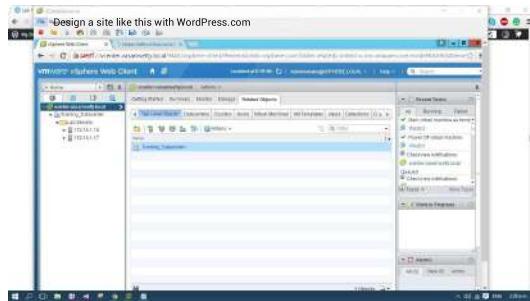


#### Creating a Template

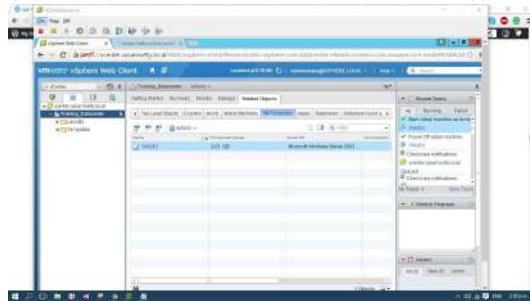
I began this task by logging into my vCenter Server via the Web Client and powering off my Windows Server 2003 virtual machine. Once that was done, I right-clicked my Windows Server 2003 virtual machine and selected 'All vCenter Actions -> Convert to Template...' from the menu. This opened up the template wizard.



I then navigated to 'vCenter -> VM Templates' and the new template showed up in the list.



I just moved my created VM into VM's folder that I had created in my previous lab.

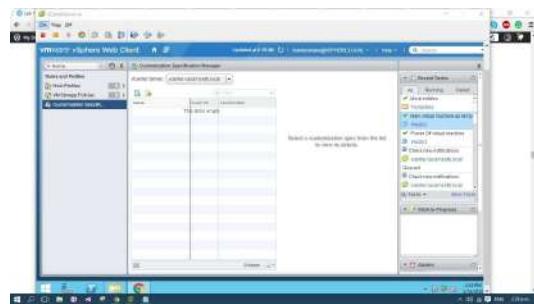


# M.SC. IT Sem 3      Server Virtualization using VMware

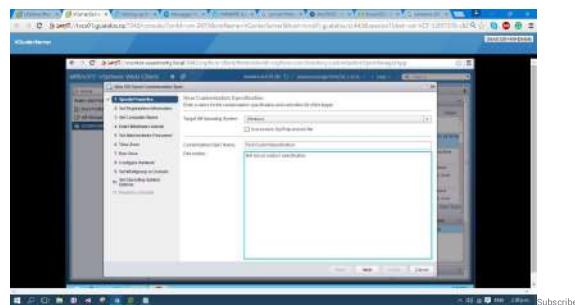
I got a template creation wizard and nished it and nally, I right-clicked the template and renamed it to 'Win2k3' as per the VMware lab instructions.

## Creating Customisation Specifications

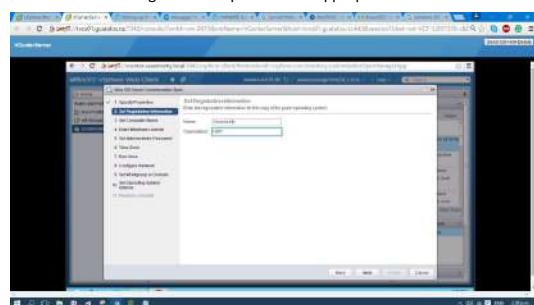
For this i navigated to the 'Customization Specification Manager' tab. I then clicked on the icon for creating a new specification.



This opened the 'New Customization Specification' wizard. For the Target VM OS type, I selected Windows. For the name, I entered 'Test-CustomSpecification' as per the lab book instructions. I also added in a description.



For the name and organisation I put in some appropriate values.

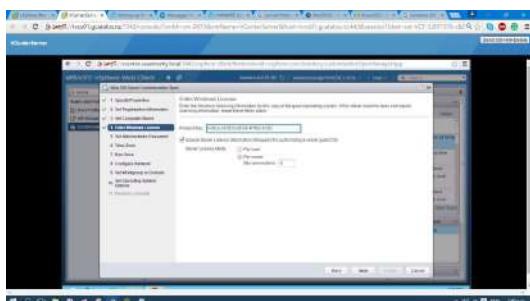


For setting the computer name, I selected the 'Use the virtual machine name' radio button and continued.

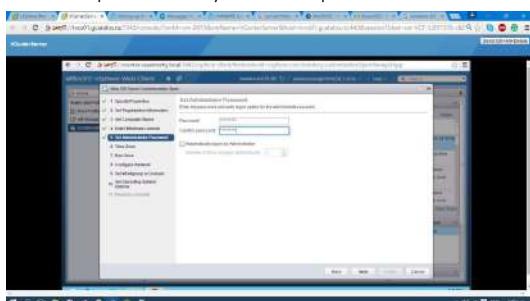


I then entered in my product key for the virtual machine's operating system and left all of the other elds as their default values.

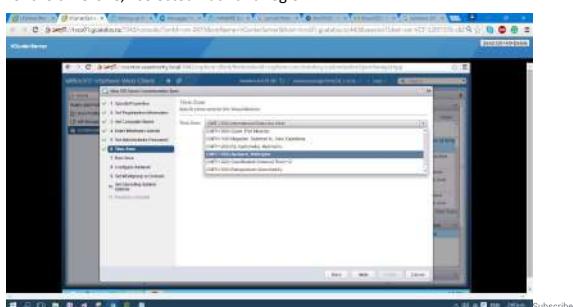
## M.SC. IT Sem 3      Server Virtualization using VMWare



I was then required to enter my administrator password for the virtual machine.

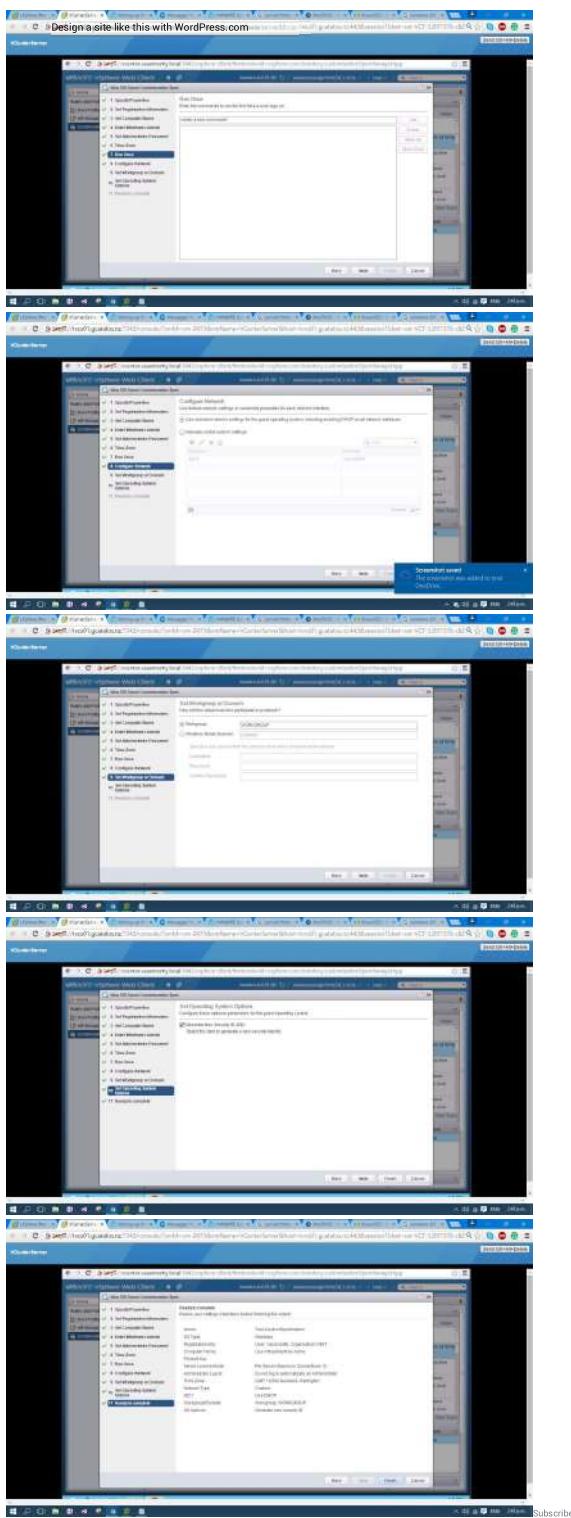


For the timezone, I selected Auckland region.



For the next four menus I left the default values and clicked 'Next'.

## M.SC. IT Sem 3      Server Virtualization using VMWare

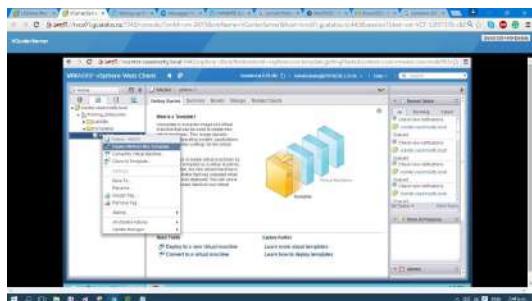


The new customisation specification then showed up in the customisation specification list.

### Deploying a Virtual Machine from a Template Design a site

like this with WordPress.com Get started

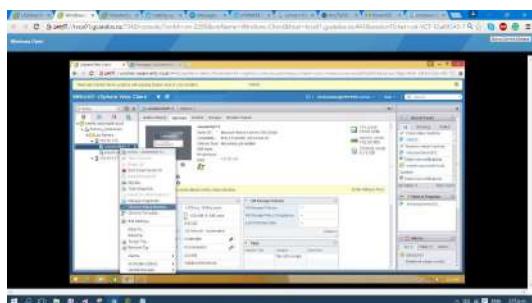
I began this task by navigating back to the 'VM Templates' section of the Web Client, right-clicking my template, and selecting 'Deploy VM from this Template'.



This allowed me once  
again in virtual machine  
con guration wizard,  
followed all the steps as  
earlier i do and nally, after  
completion i found a  
running VM I found that  
both of them were set to  
'Obtain an IP address  
automatically' so that  
wasn't the problem. I then  
checked the computer  
name for each computer  
and found that both of  
them were still using the  
original 'vasamsetty'  
name. To x this, I changed  
both of the computer  
names on both of the  
virtual machines to their  
respective virtual machine  
names



**Cloning a Virtual Machine that is Powered On**  
I started this task by  
navigating back to my  
vSphere Web Client,  
navigating to 'VMs and  
Templates', rightclicking  
on my 'vasamsetty01-2'  
virtual machine, and  
selecting 'Clone to Virtual  
Machine...'.



I followed the setup wizard, but  
unfortunately i had't had enough  
space to continue my installation. So,  
I interrupted my installation but I  
learnt how to clone a virtual machine  
while am testing on my main gcatalogos  
Subscribe network. This make me  
enough con dent in creating clones.

## Practical 9: Modifying the Virtual Machine

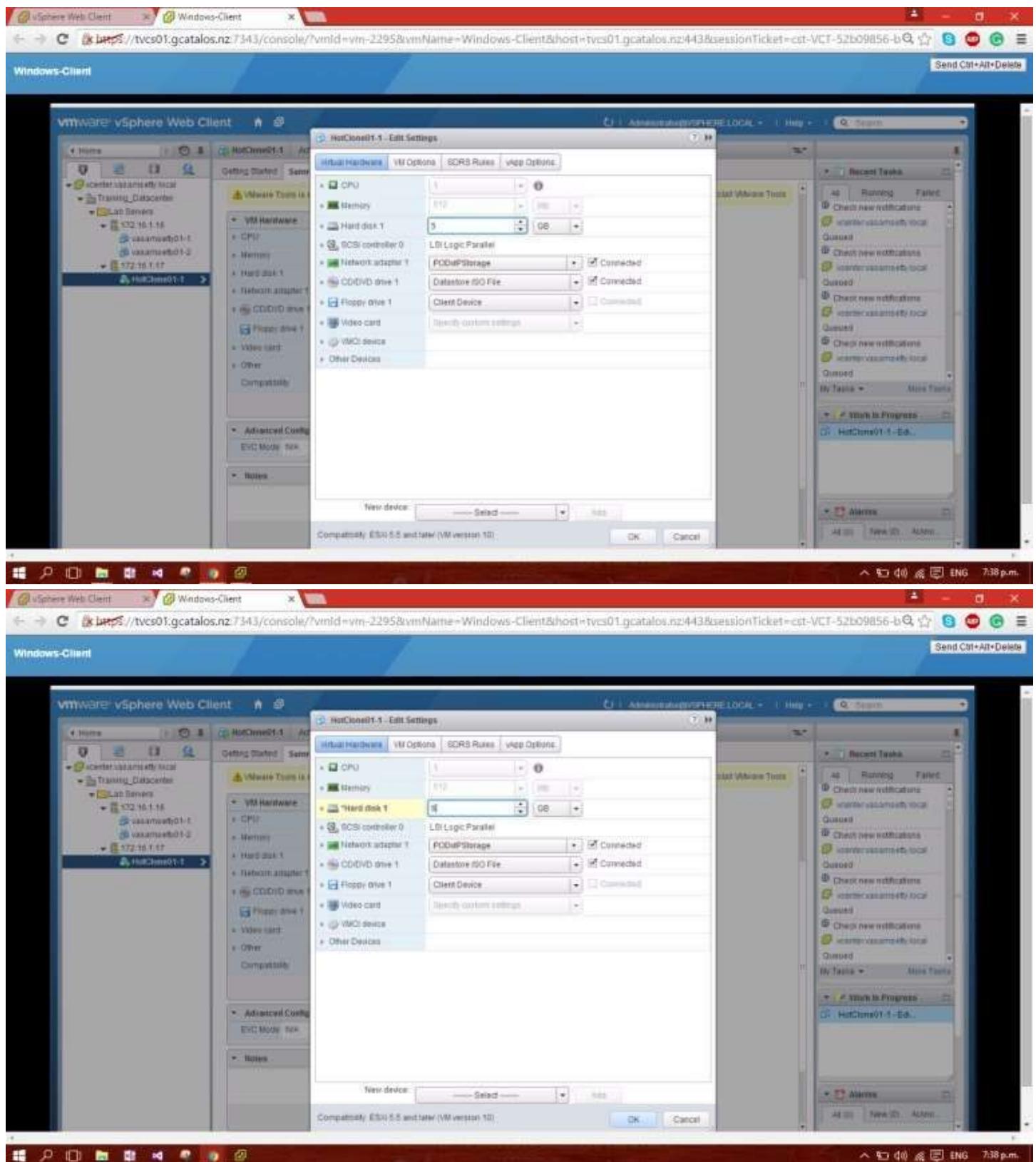
In this lab ,

I was required to do the following tasks:

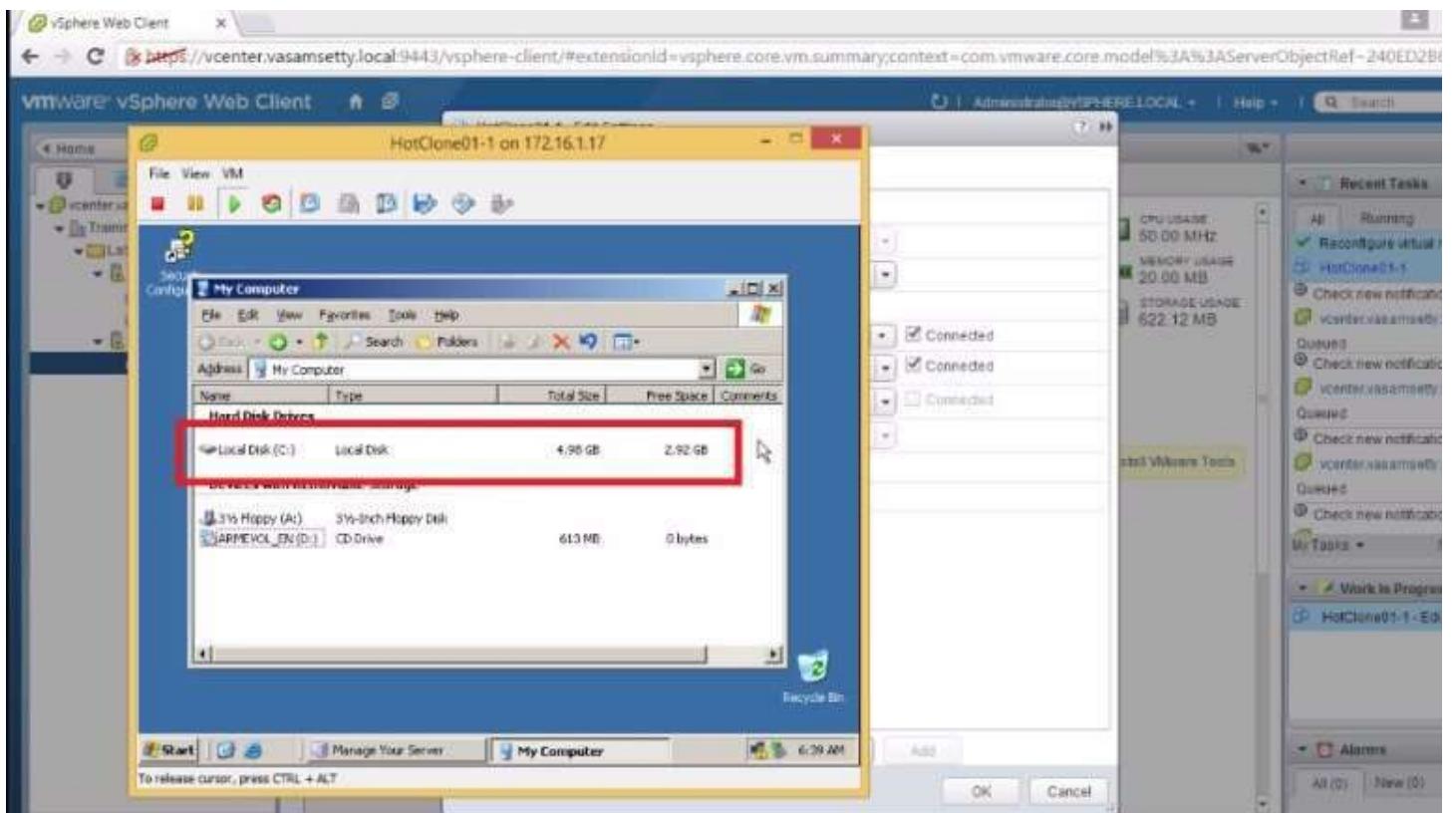
- Increase the Size of a VMDK File
- Adjust Memory Allocation on a Virtual Machine
- Rename a Virtual Machine in the vCenter Server Inventory
- Add a Raw LUN to a Virtual Machine
- Expand a Thin-Provisioned Virtual Disk

Increasing the Size of a VMDK File

For this task i navigated to Vcenter>Host and Clusters selected my second esxi host. From there I right clicked on my HotClone01-1 virtual machine and selected Edit Se ings . We will get a window pop-up showing Edit se ings menu. I changed my hard disk size from 5 GB to 6 GB. My Lab instructions values are different but ultimately we need to change the hard disk size.



Unfortunately i can't see any increase in size on my virtual machine hard disk size. The total hard disk size in my windows server is still 5 GB.

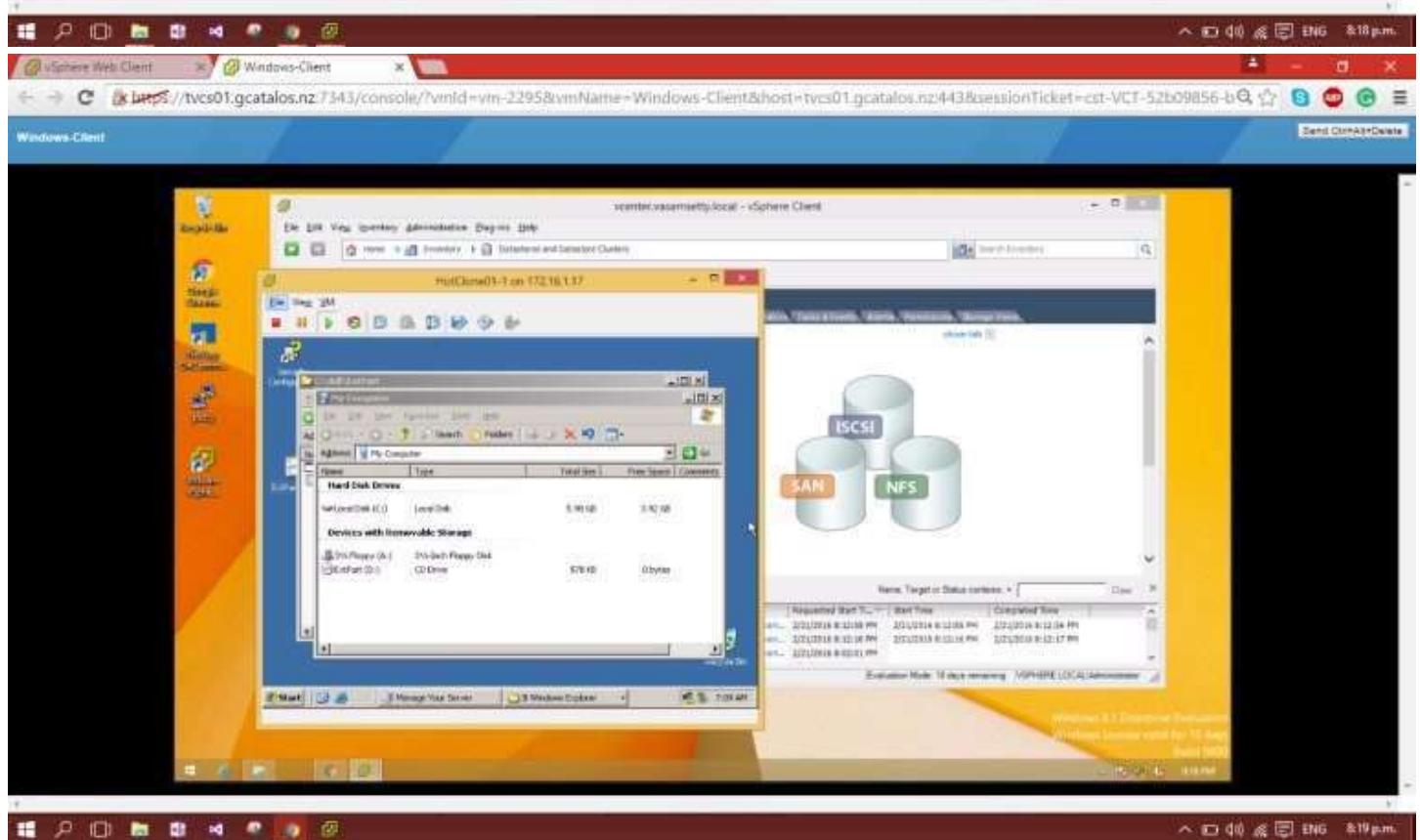
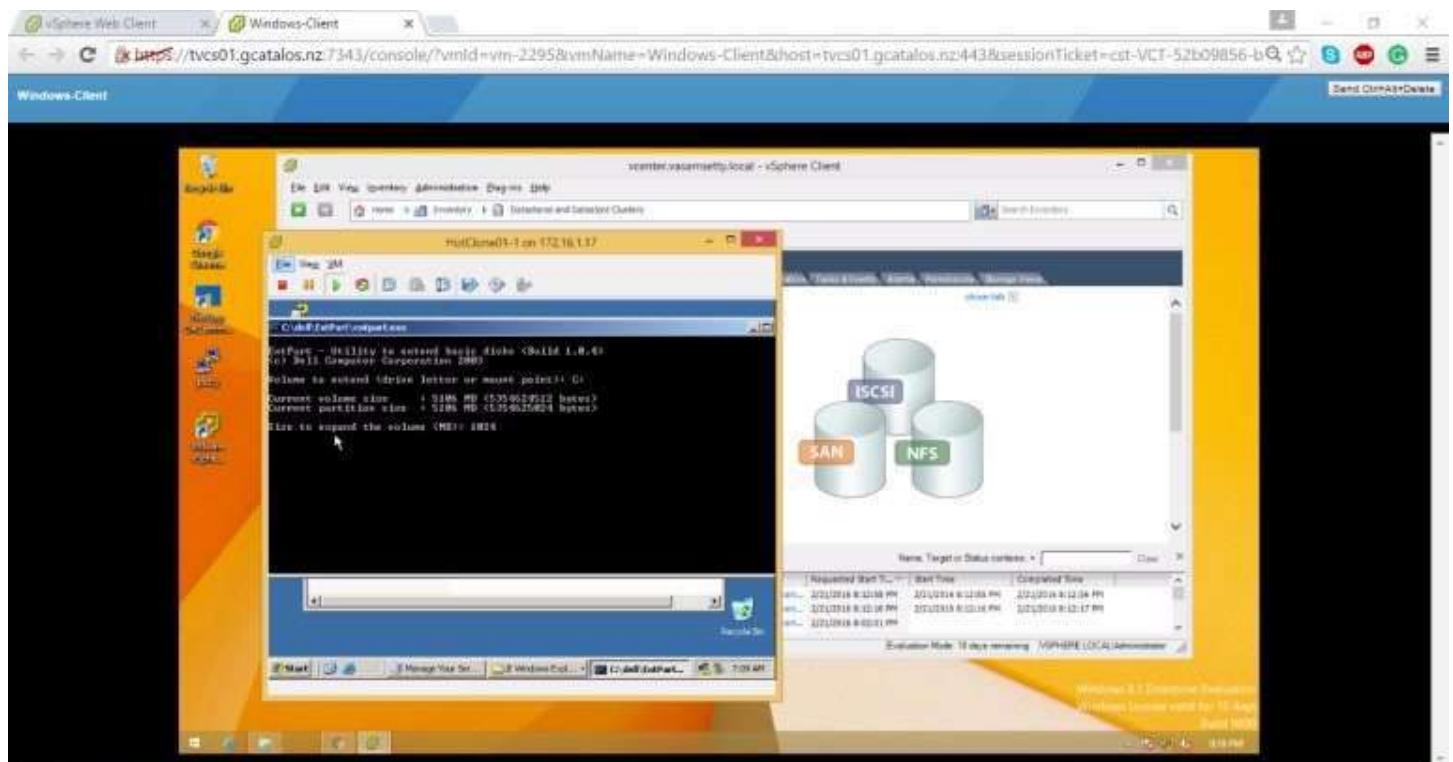


In order to expand the drive to accommodate the extra gigabyte that I defined, I needed to download 'extpart'. This is because there are no native applications on Windows Server 2003 that allow for the extension of partitions. For this we need to open a internet connection from your virtual machine goto your Internet Explorer and downloaded 'extpart' from: <http://download.dell.com/app/ExtPart.exe> (<http://download.dell.com/app/ExtPart.exe>) .

**External Work:** I tried hard for this but i failed to create an internet connection. In the second try i downloaded the respective files on my local browser,grouped the files into an ISO-1 and copied the ISO-1 into another folder and created another ISO-2 file and I mounted to my windows client(to tallos network) which made me available to copy ISO-1 files in my windows client VM.

In my client-OS i copied ISO-1 files to desktop and opened my FQDN:vcenter.vasamse .local network and mounted the ISO-1 file. From this task i can able to view my necessary files on my created VM.

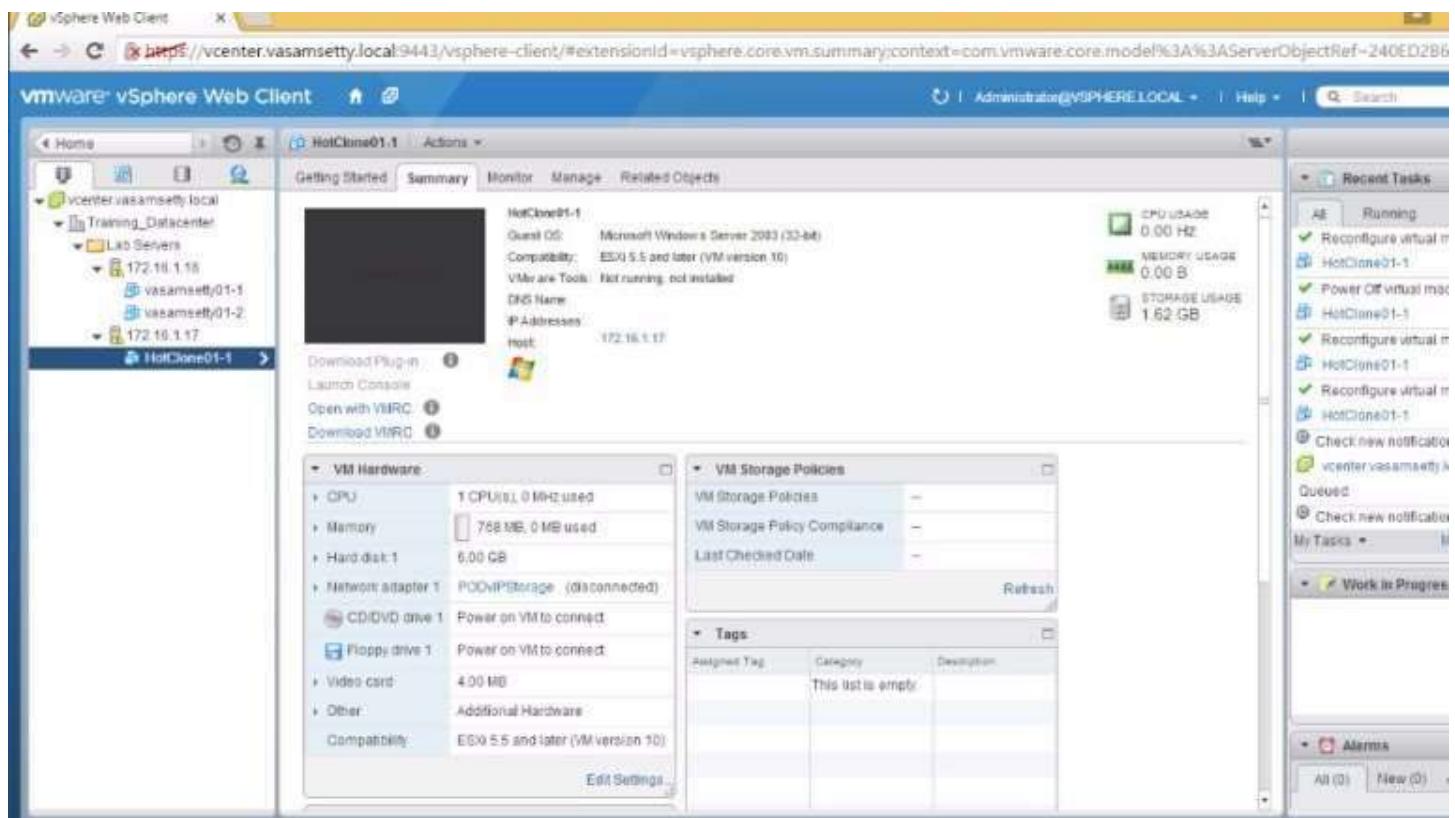
So, now lets come back to our post again. As in this post finally i copied ExtPart.exe file on my Win2003 server and runned the exe, entered in "C:" as the drive to change, and entered "1024" as the amount (in MB) to extend the drive by. YUP i can see the hard disk size change



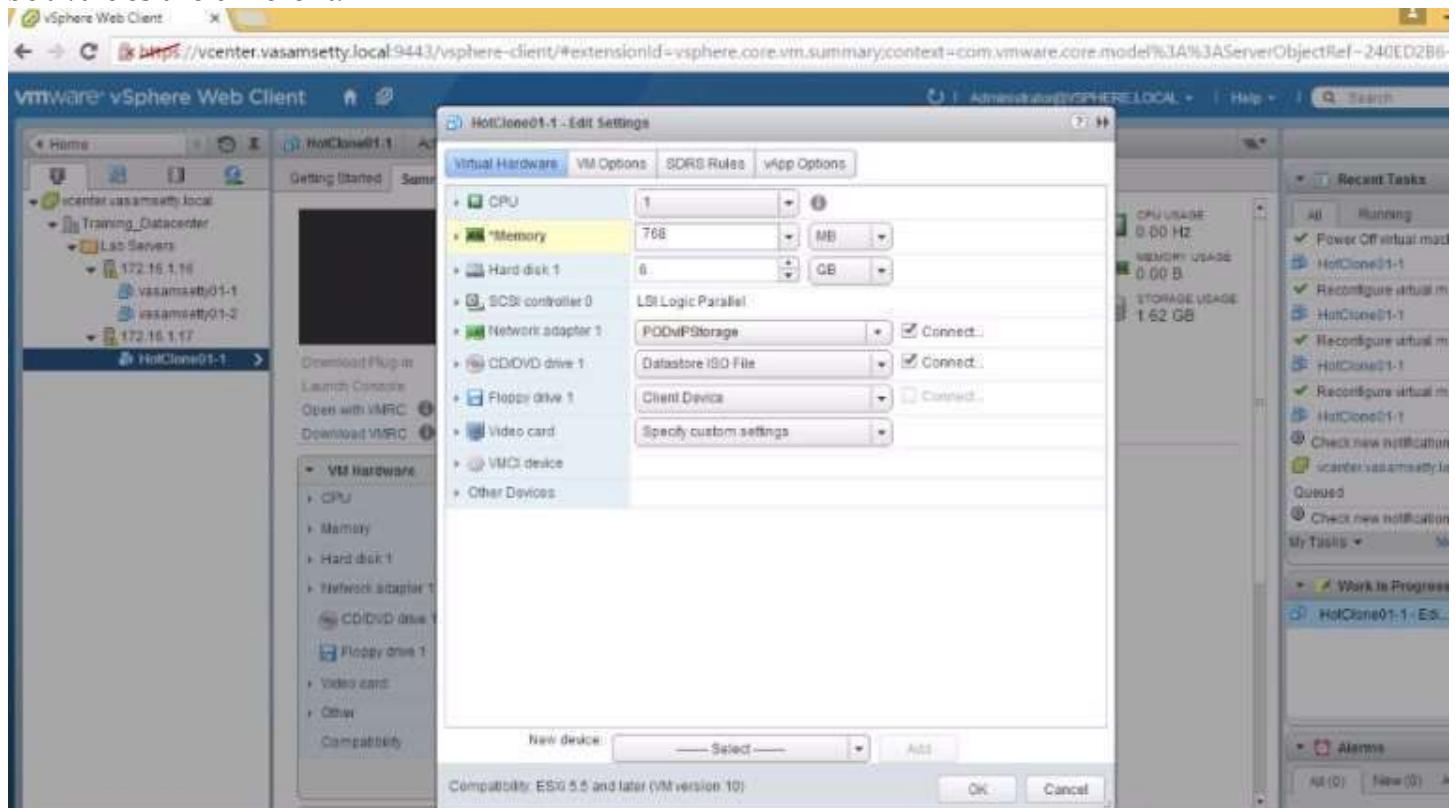
Previously it showed 4.98 GB total size whereas now it shows 5.98 GB total size.

#### Adjusting Memory Allocation on a Virtual Machine

To start this task, I shut down my Hot-Clone01 virtual machine, switched back to the vSphere Web Client, right-clicked the Hot-Clone01 virtual machine and selected 'Edit Se 旳@ings...'.



From here I changed the memory amount to 768 MB. My lab instructions says that to increase the value but values are different.



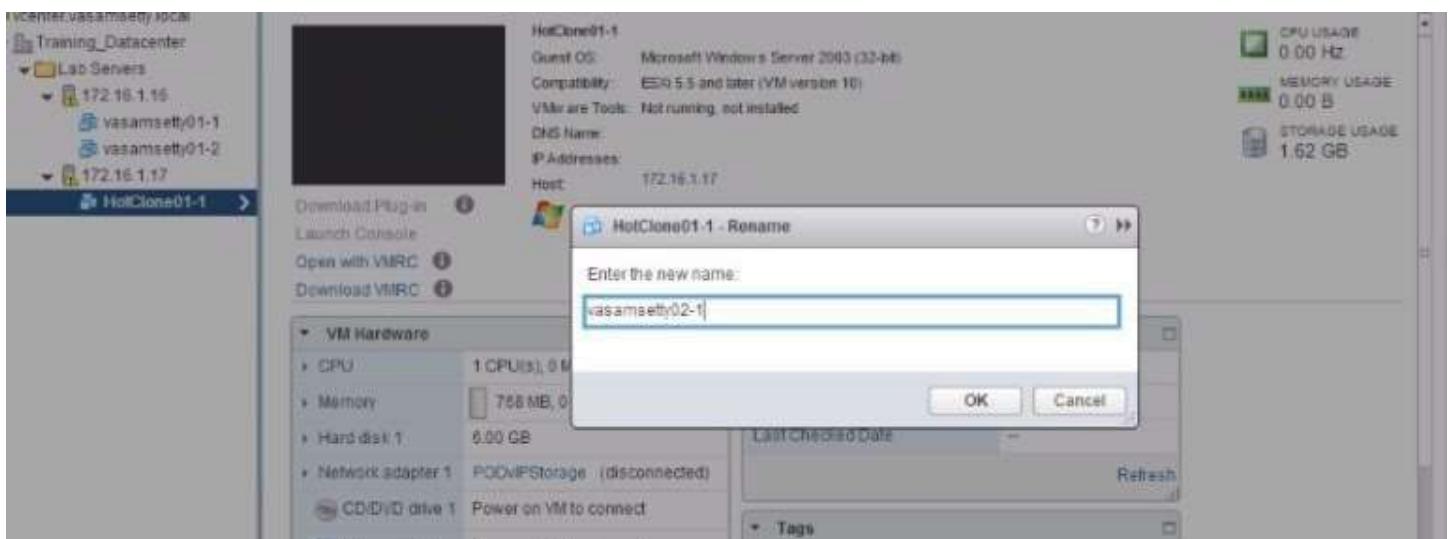
After making the change, I clicked on the Hot-Clone01 virtual machine once again and then navigated to the 'Summary' tab. From here I confirmed that the memory had increased in size.

The screenshot shows the vSphere Web Client interface. On the left, the inventory tree shows a folder named 'Training\_Datacenter' containing several hosts and a virtual machine named 'HotClone01-1'. The central area displays a summary card for 'HotClone01-1', which is currently powered off. The card provides details such as Guest OS (Microsoft Windows Server 2003 (32-bit)), Compatibility (ESX 5.5 and later (VM version 10)), and IP Addresses (172.16.1.17). To the right of the summary card are three performance graphs: CPU usage (0.00 Hz), Memory usage (0.00 B), and Storage usage (1.62 GB). Below the summary card, there are tabs for 'Getting Started', 'Summary', 'Monitor', 'Manage', and 'Related Objects'. The 'Manage' tab is selected. The right side of the interface features a 'Recent Tasks' panel listing various administrative actions, such as 'Reconfigure virtual machine' and 'Power Off virtual machine', along with their status (Running or Queued).

### Renaming a Virtual Machine in the vCenter Server Inventory

To begin this task, I right-clicked my Hot-Clone01 virtual machine and selected “Rename...”.

This screenshot shows the same vSphere Web Client interface as the previous one, but with a context menu open over the 'HotClone01-1' virtual machine. The menu includes options like 'Open Console', 'Power On', 'Move To...', and 'Rename...'. The 'Rename...' option is highlighted with a blue selection bar. The rest of the interface remains consistent with the first screenshot, showing the inventory tree, summary card, and recent tasks panel.

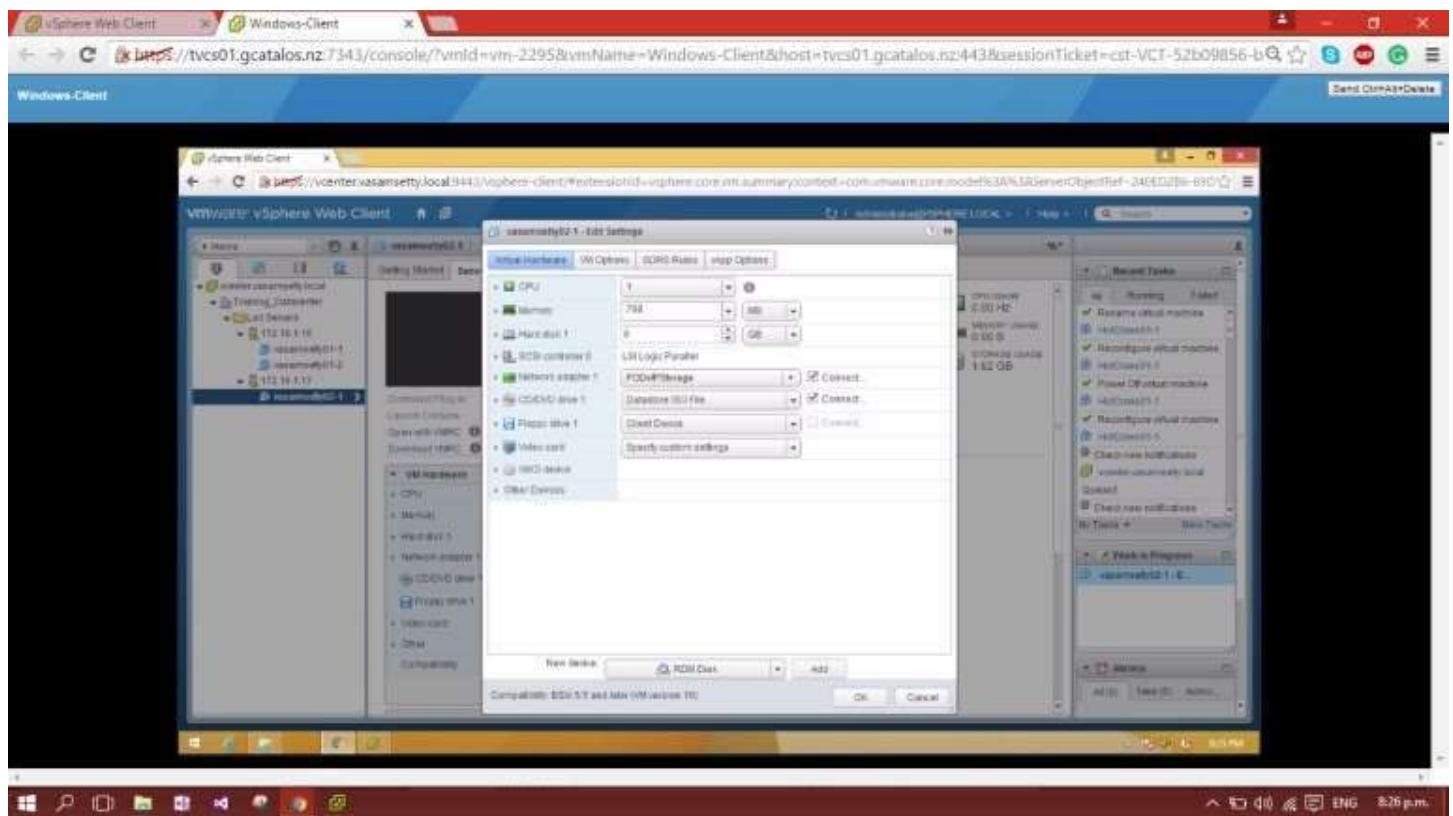


The renamed virtual machine can be seen below:

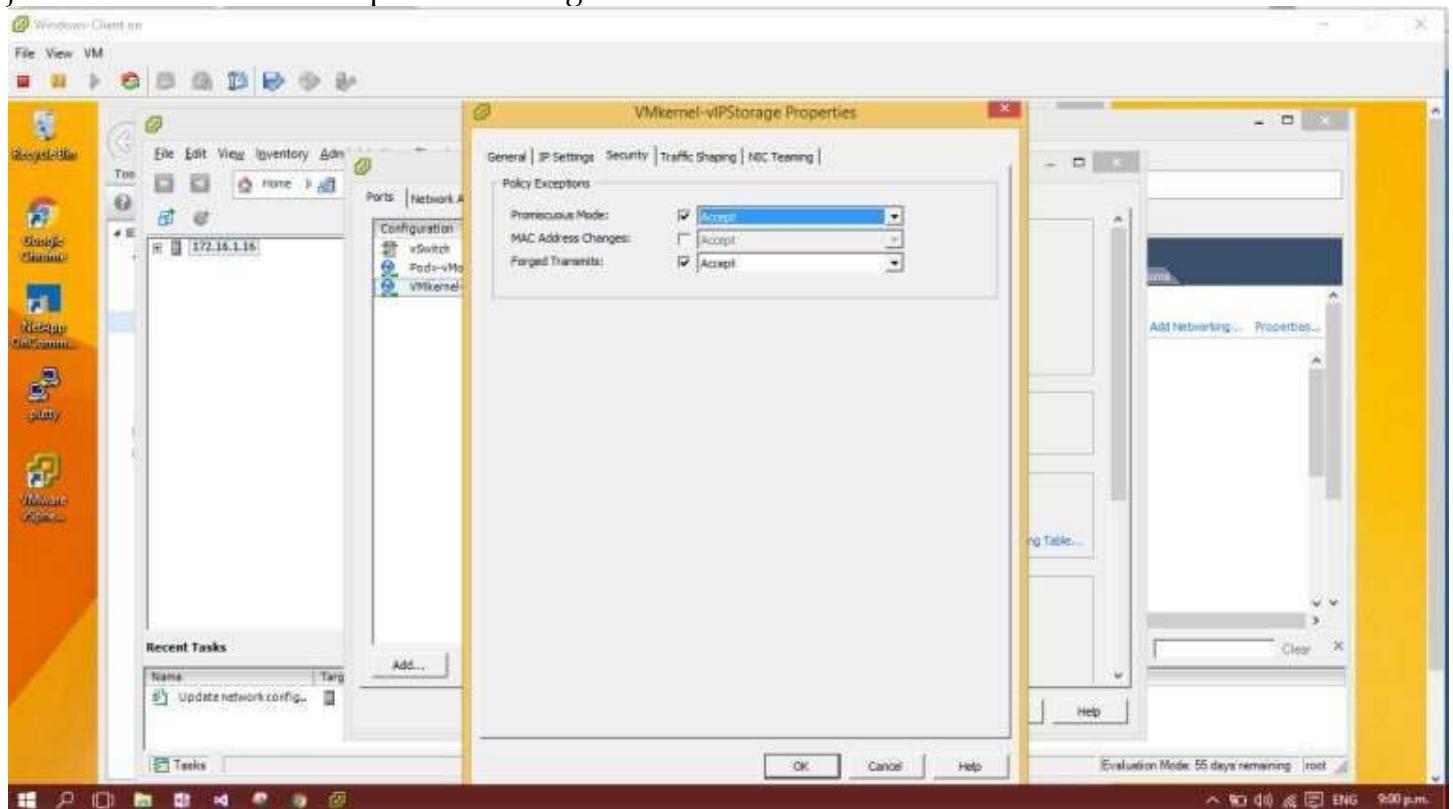
This screenshot shows the VMware vSphere Web Client interface. The left sidebar shows the same organizational structure as the desktop client: a datacenter, a lab server folder, and a host. The main pane displays the summary details for the virtual machine 'vasamsetty02-1'. The summary card shows the guest OS as Microsoft Windows Server 2003 (32-bit), compatibility as ESXi 5.5 and later (VM version 10), and the host as 172.16.1.17. Below the summary card, the 'VM Hardware' section lists components like CPU, Memory, Hard disk 1, Network adapter 1, and CD/DVD drive 1. To the right, sections for 'VM Storage Policies' and 'Tags' are visible.

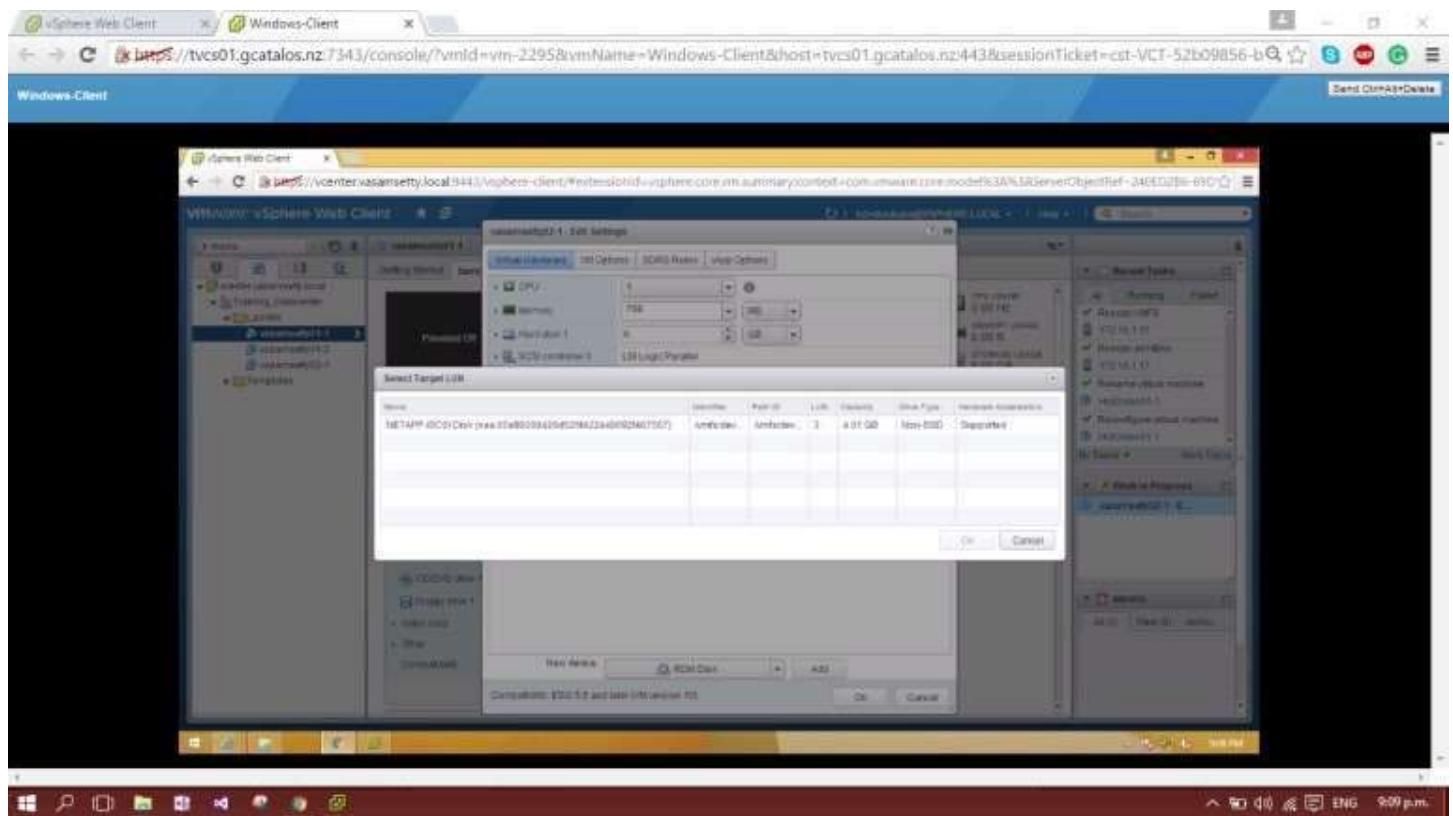
### Adding a Raw LUN to a Virtual Machine

To start this task, I right-clicked my 'Jamie01-2' virtual machine and selected 'Edit Settings...'. From the 'New device' area, I selected RDM Disk and clicked Add.

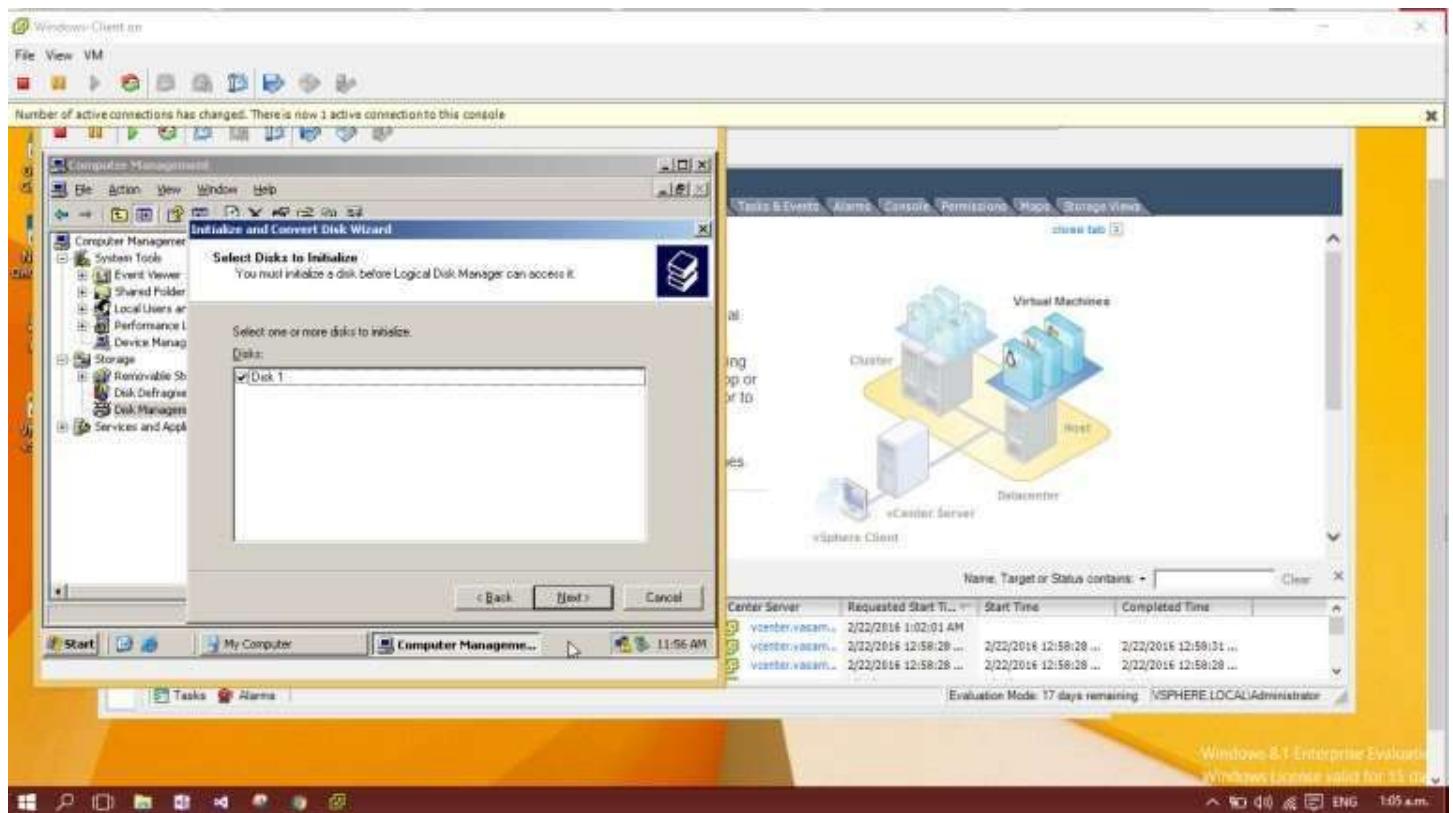


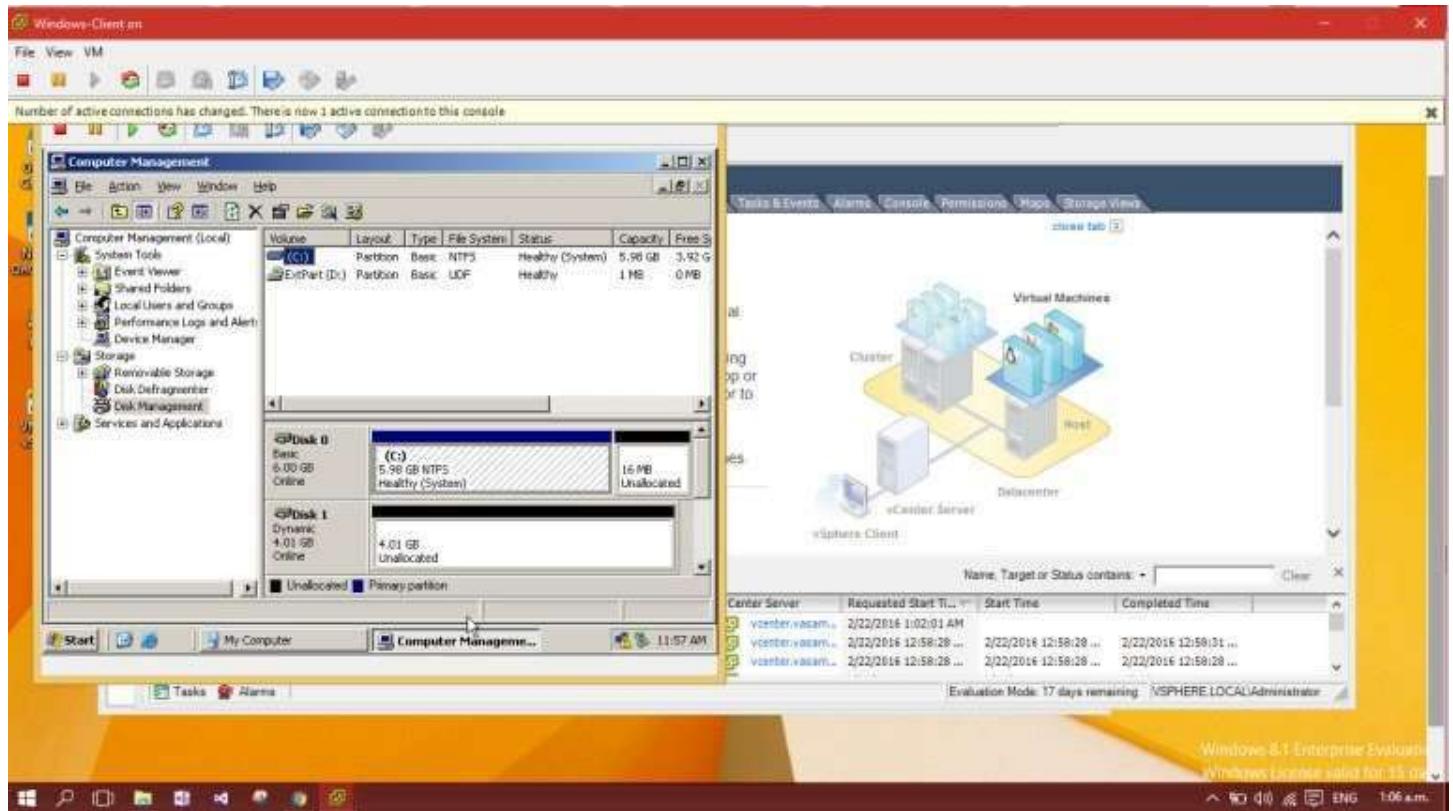
I just tried to add Raw Disk and found no LUN's available at that time. This is because I just skipped a step in the previous lab to enable Promiscuous Mode or Forged Transmits on all of my vSwitches, So i just done it before this step and it went good I can see the new LUN now when i click on add.





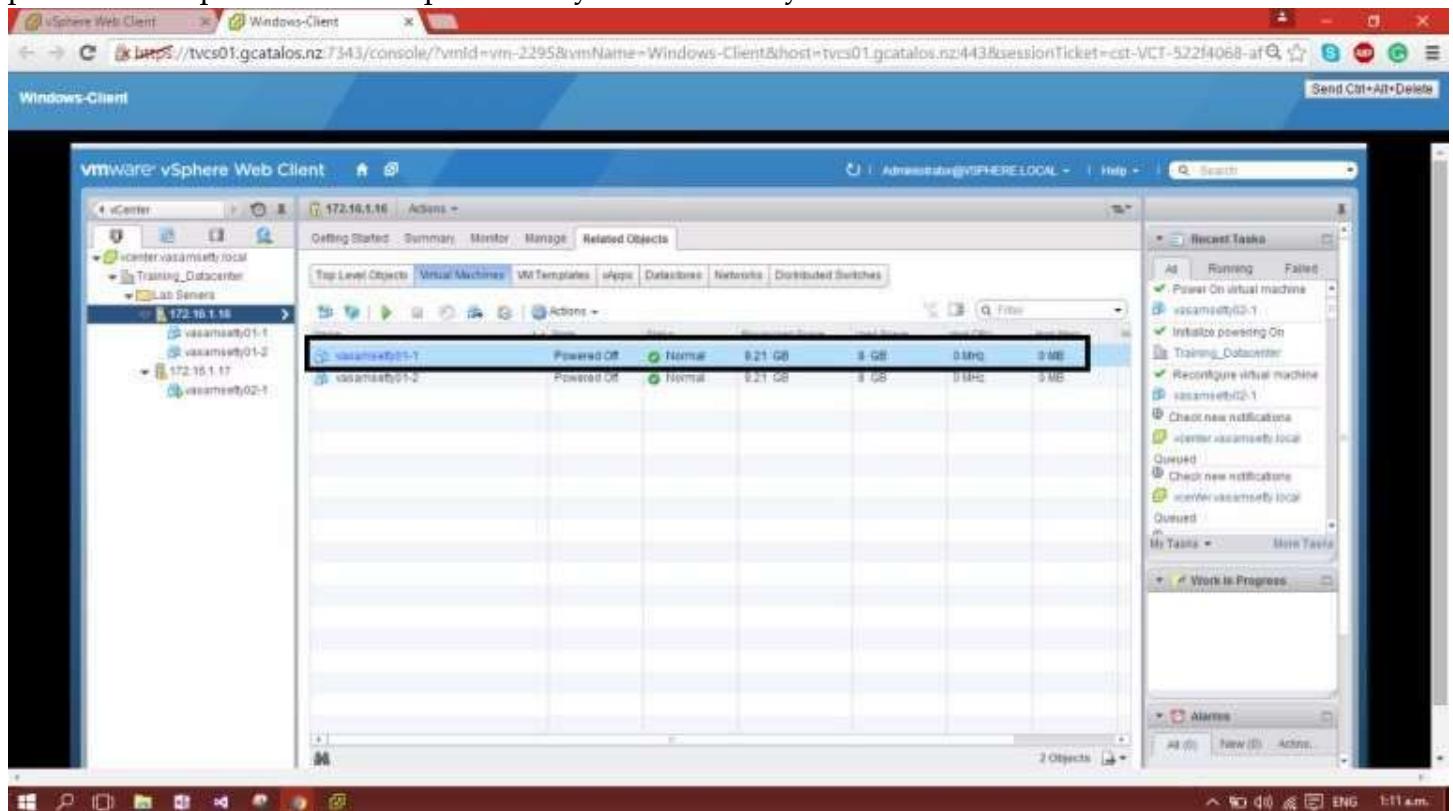
Successfully a raw disk is added to my virtual machine . I just logged into my VM and to see that whether the disk is initialized if not am gonna initialize it. To initialize goto Computer Management>Disk Management>Run the wizard that appears,I could see that the disk showed up as 'Disk 1'.





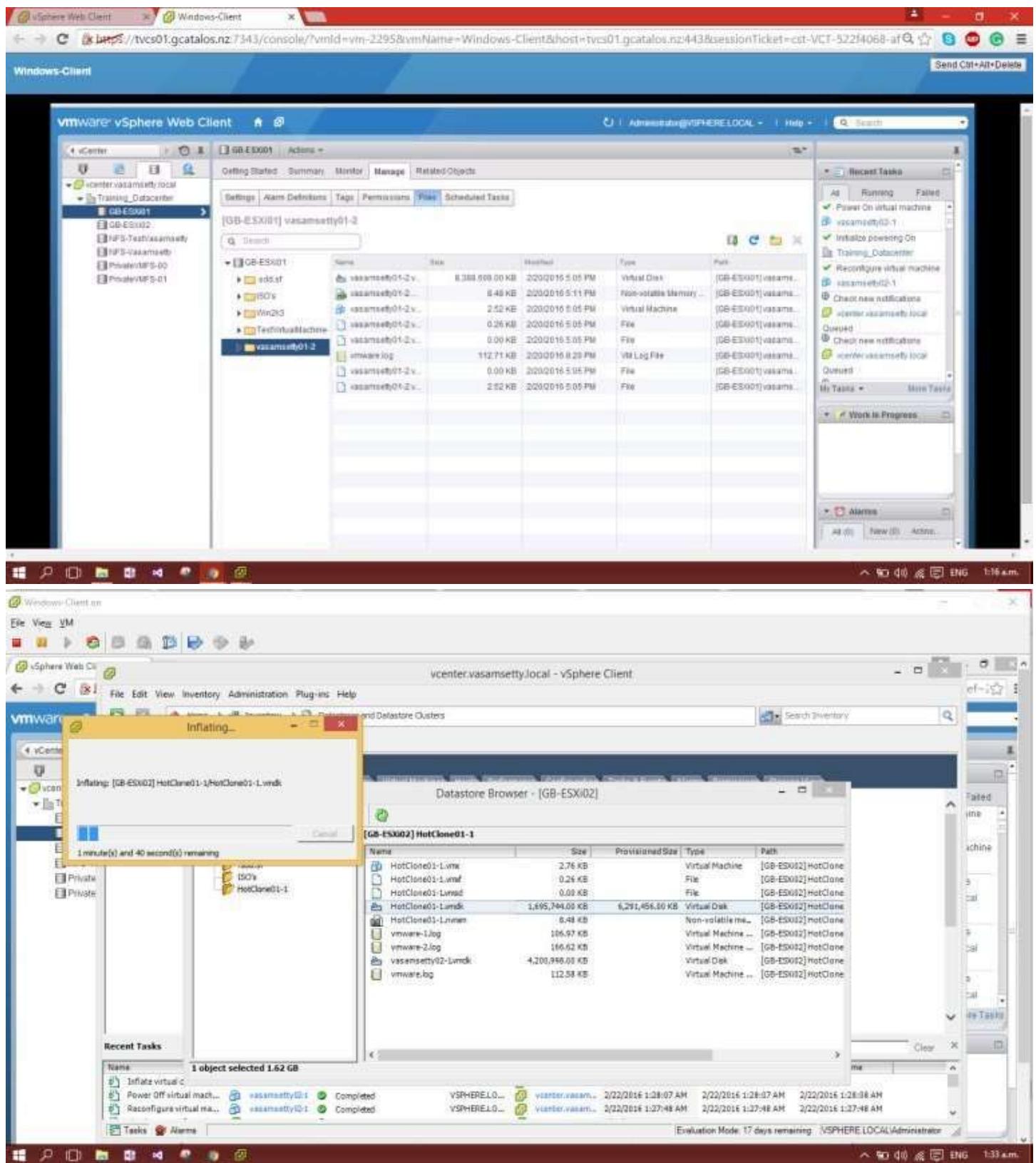
### Expanding a Thin-Provisioned Virtual Disk

To begin this task, I navigated to 'Home -> Hosts and Clusters -> Lab Servers -> <TheIPofMyFirstESXiHost> -> Related Objects -> Virtual Machines' and reviewed the amount of provisioned space and used space for my 'vasamse 仸@y01-1' virtual machine.

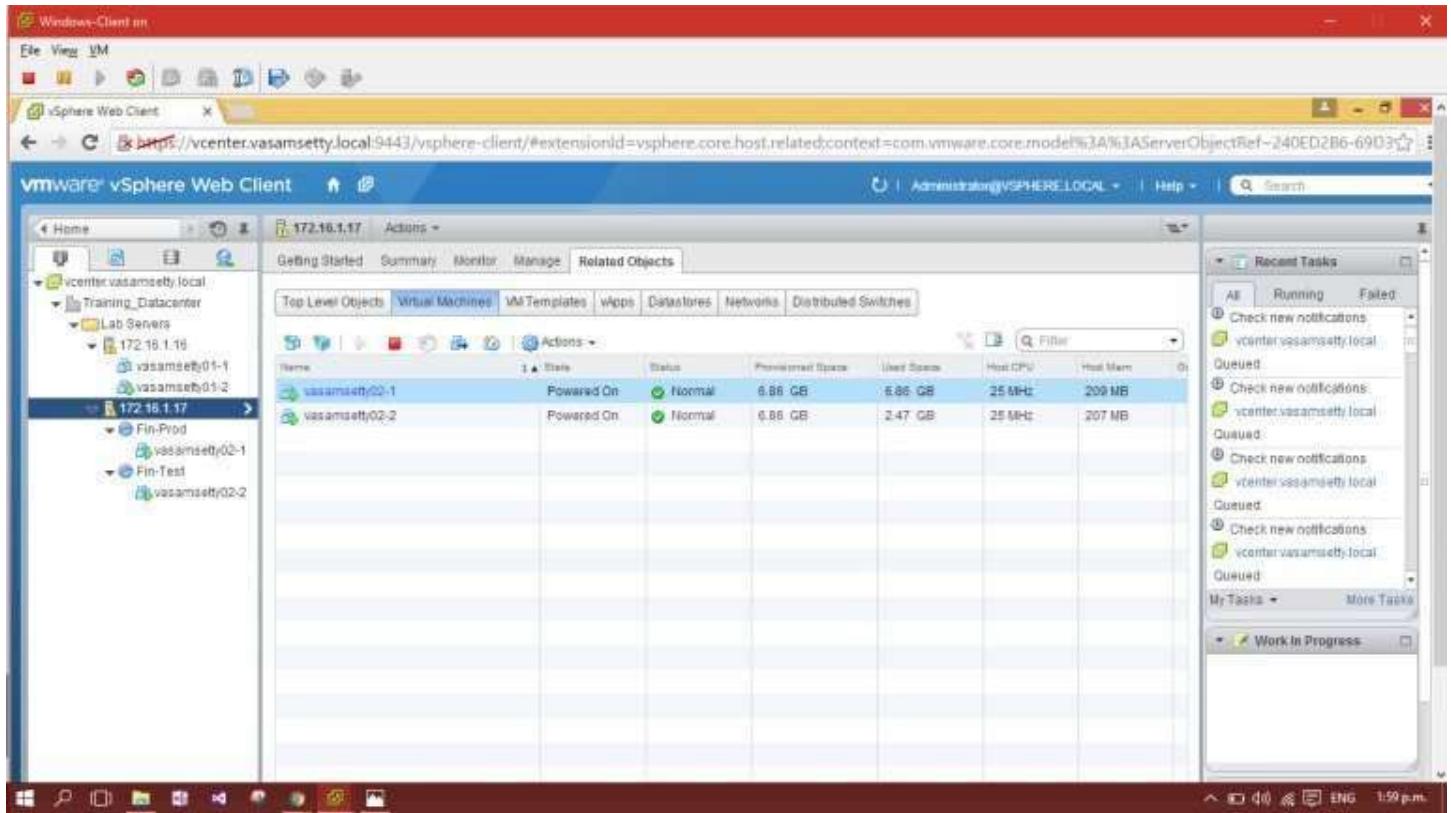


In the next step i clicked on my selected VM to check the hard drive details and noted which datastore is on use and found that it was on my first esxi datastore.I then shut down the virtual machine and navigated to 'Home -> Storage' where I right-clicked my Local-ESXi01 datastore and selected 'Browse Files'.

From there i had gone to my virtual machine vasamse 仮仮y01-2 and right clicked on vasamse 仮仮y01-2 and clicked on inflate. Simillary i did the same process for the another virtual machine virrtual machine02-1



After completing the inflation for the new hard drive, I navigated to 'Home -> vCenter -> Hosts and Clusters -> Lab Servers -> <TheIPofMyFirstESXiHost>', clicked the Related Objects tab, clicked on the Virtual Machines tab, and powered on my 'vasamsetty02-2' host. I paid particular attention to the 'Used Space' and 'Provisioned Space' columns and found that due to the inflation, all of the available disk space was now being used.



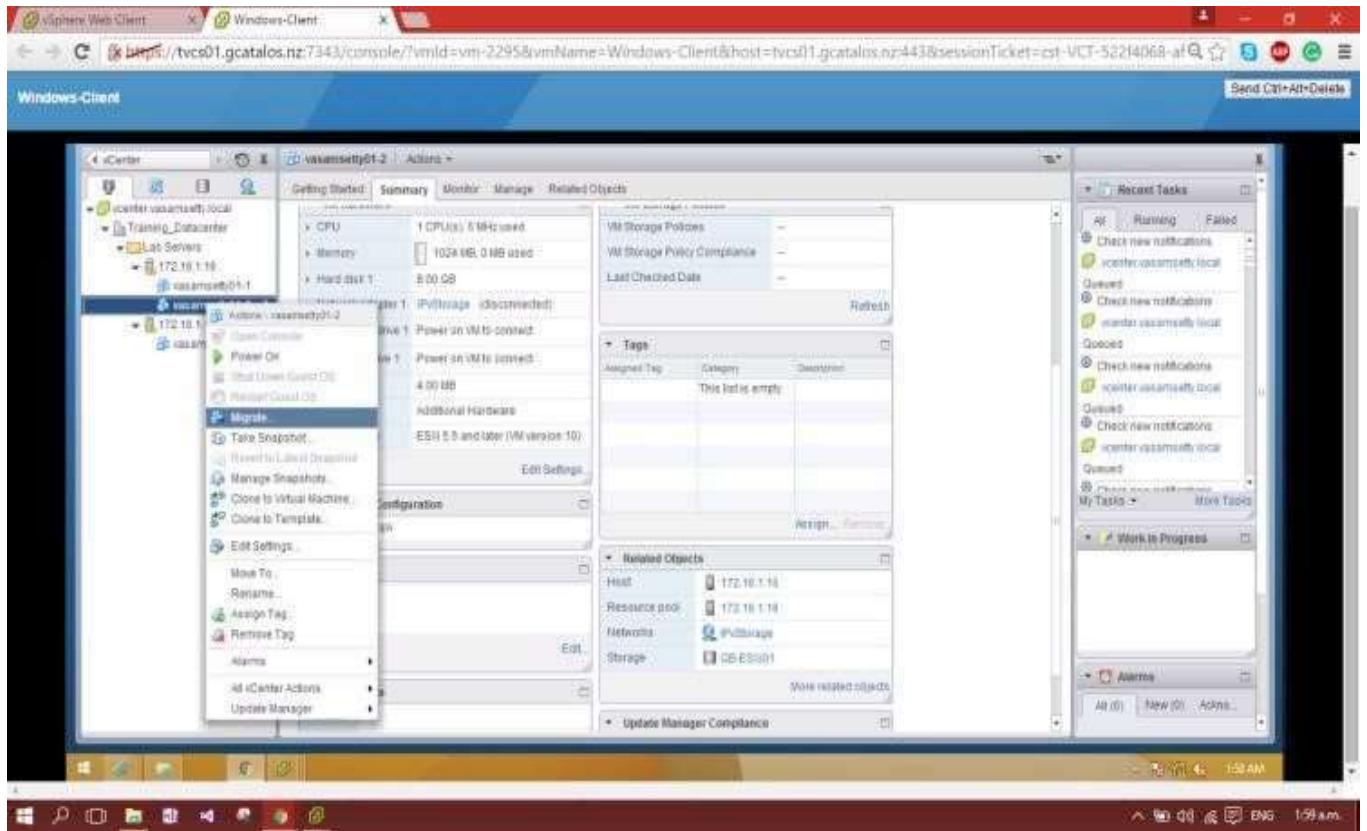
[Blog at WordPress.com.](#)

## Practical No 10:Migrating Virtual Machines

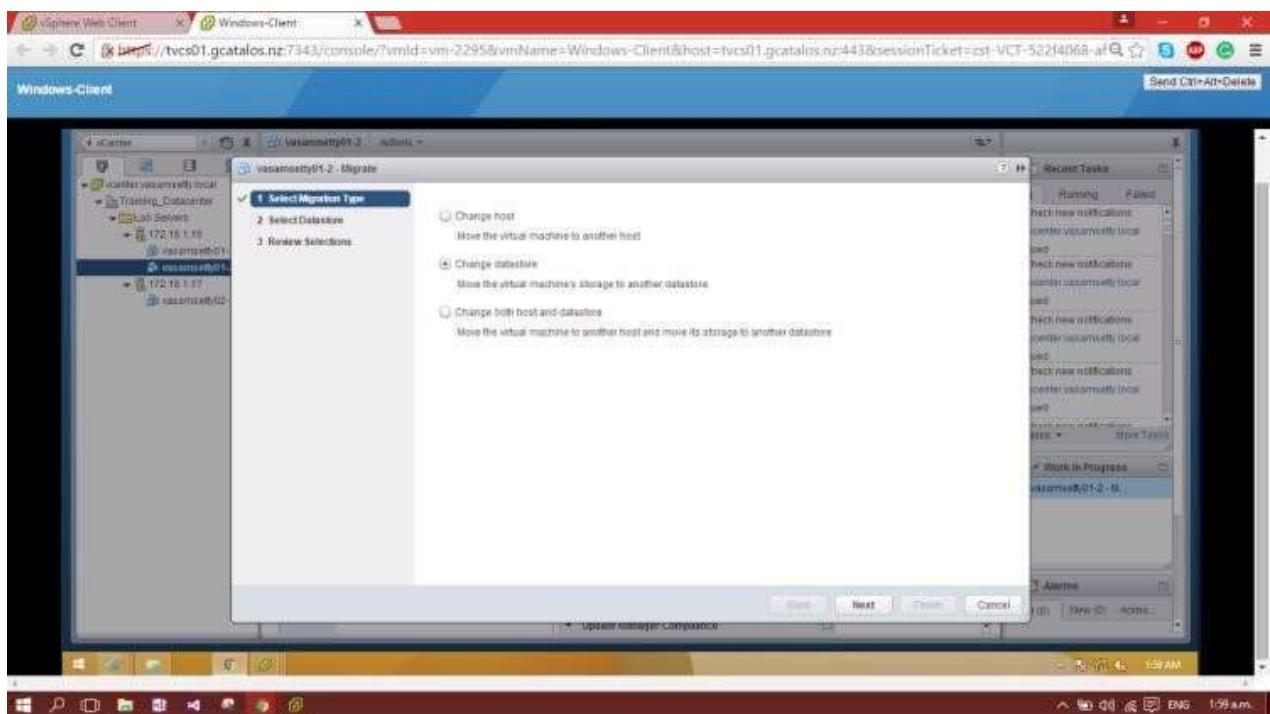
### Migrate Virtual Machines Files from Local Storage to Shared Storage

#### Migrating Virtual Machine Files with vSphere Storage vMotion

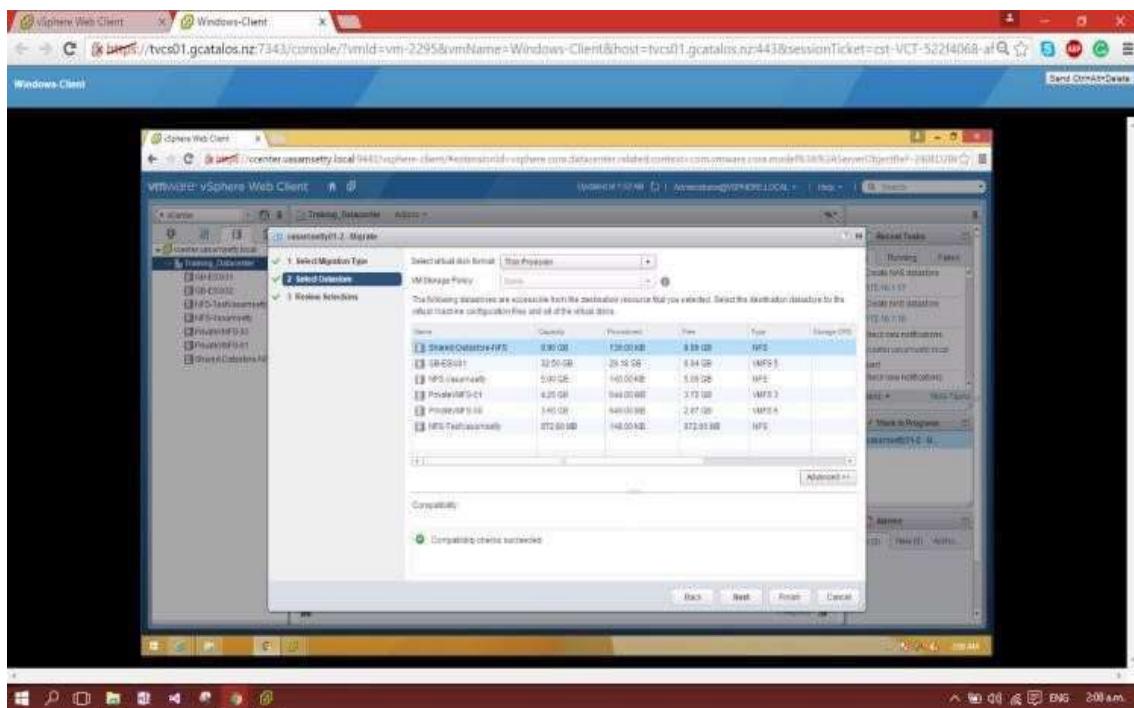
For migrating virtual machine files with vsphere storage vMotion. I navigated vasamsetty01-2 virtual machine and right clicked and selected migrate. Before that i noted down in which datastore my vm exists its on GB-ESXi01



I then selected the 'Change Datastore' radio button (although I didn't really have a choice).



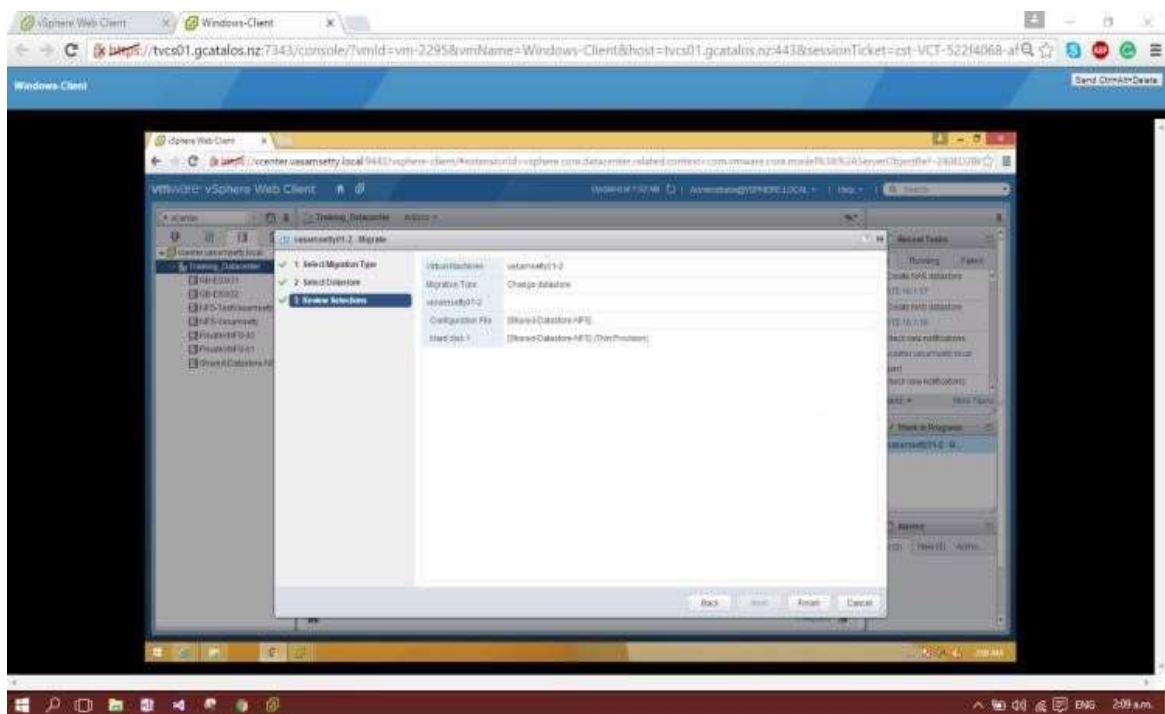
According to my lab manual i need to change the datastore from previous datastore to the shared datastore. Since my disk space's of datastores is not sufficient i created a new datastore named Shared Datastore with disk space 20 gb and initiated my change of datastore to the share datastore.



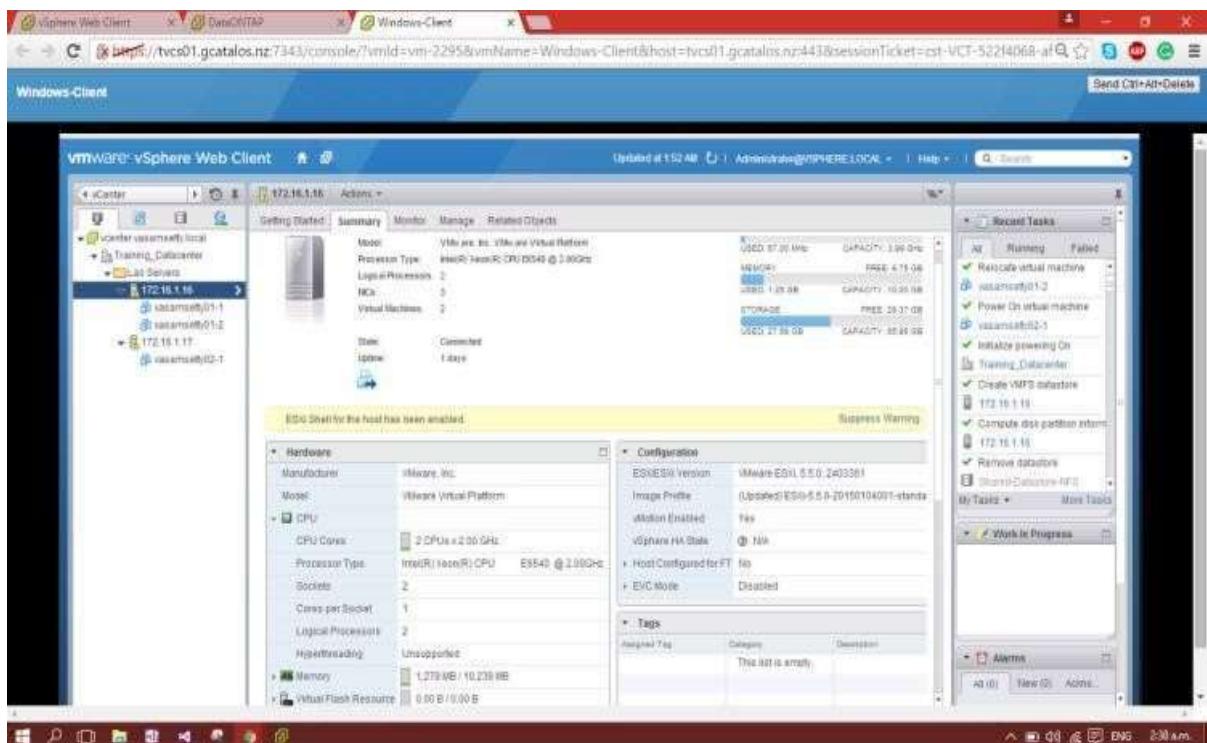
After making my selections, I reviewed the settings and confirmed the migration.

## M.SC. IT Sem 3

## Server Virtualization using VMWare

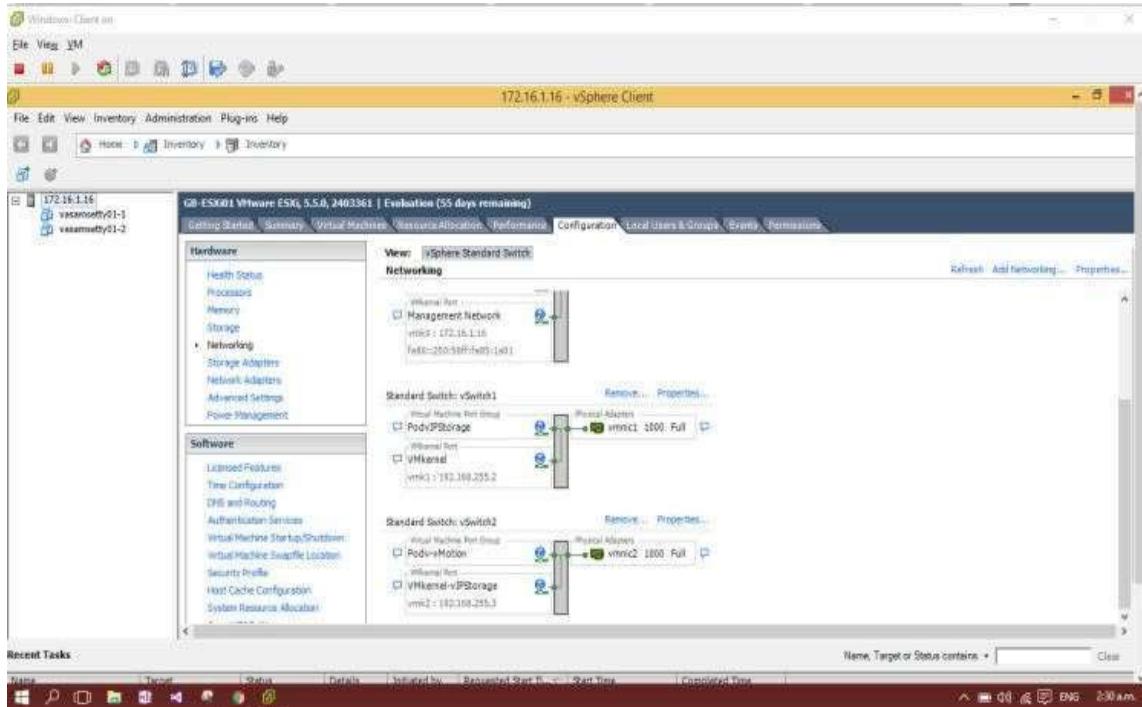


In this image I found that I unable to see the related objects. but in the recent tasks you can see my relocation done successful.

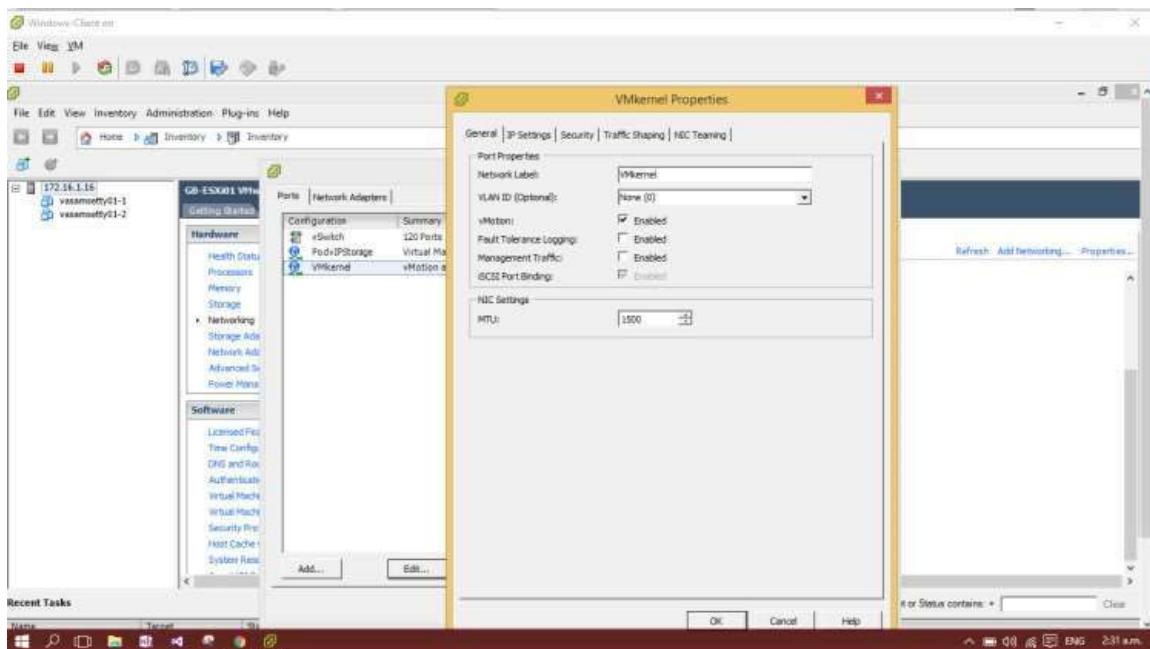


### Creating a Virtual Switch and a VMkernel Port Group for vSphere vMotion Migration

I think i had done all the tasks for this in my lab 8 and lab 9



I did have to change one setting however. I opened up the vSphere Client, logged in to my first ESXi host, navigated to 'Networking -> Properties...(for my vSwitch2) -> Selected the VMkernel -> Edit...' and made sure the vMotion tick box was ticked.



I also repeated this step for my other ESXi host.

### Verifying That My ESXi Host Meets vSphere vMotion Requirements

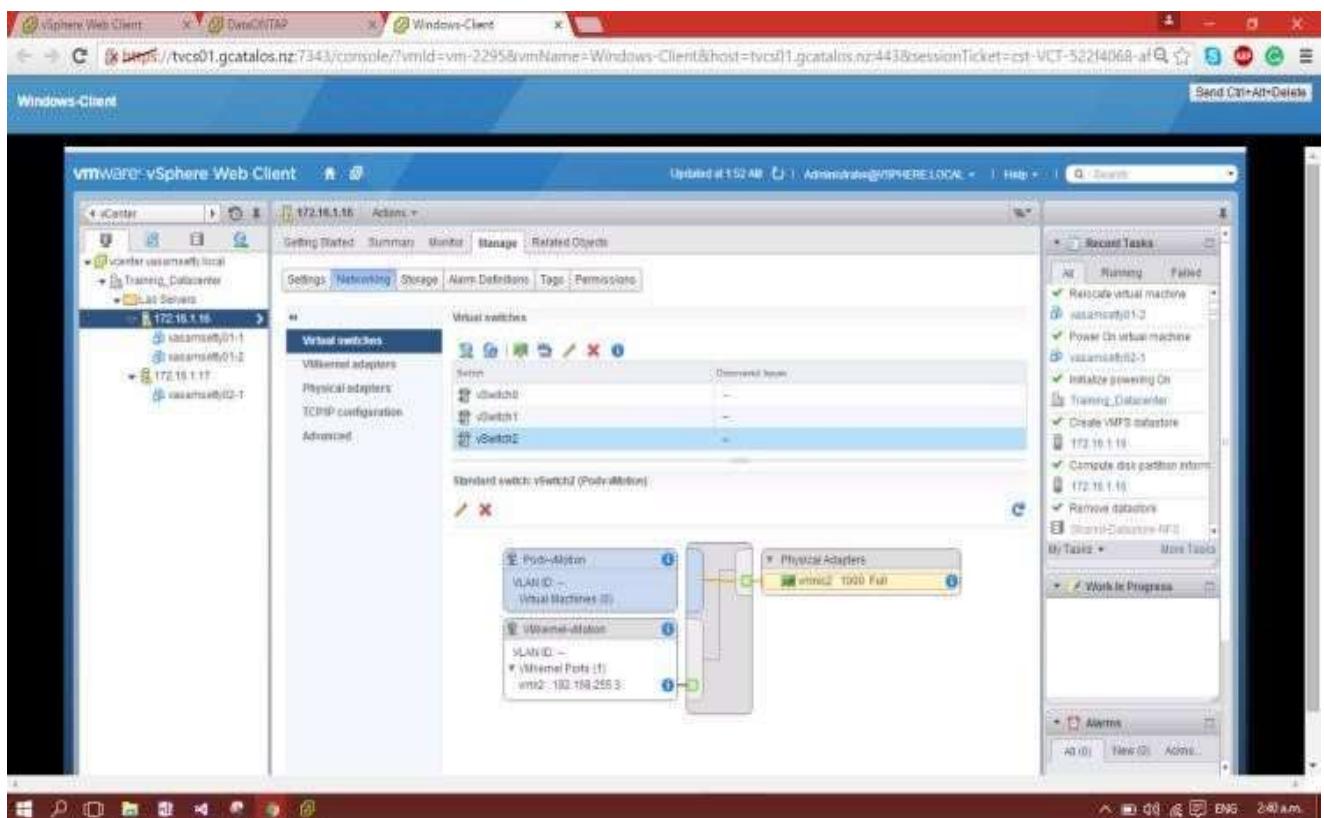
To start this task, I navigated to 'Home -> vCenter -> Hosts and Clusters', selected each of my ESXi hosts, and reviewed that their CPUs were compatible under the 'Summary' tab. I assume that they are both compatible as I didn't see anything to say that they weren't. ESXi host 1's Processor Type information:

The screenshot shows the VMware vSphere Web Client interface. In the left sidebar, the navigation tree includes 'vCenter', 'Hosts and Clusters', and '172.16.1.16'. The main content area displays the 'Summary' tab for the host '172.16.1.16'. Key details shown include:

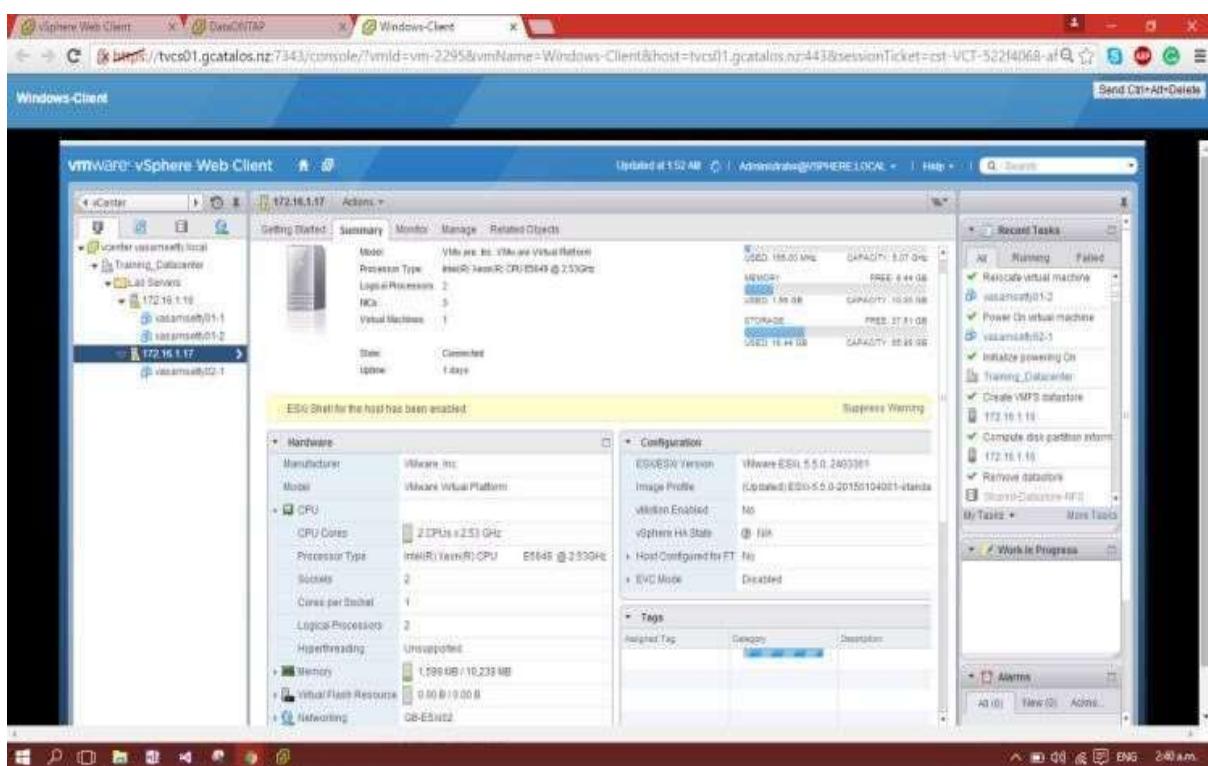
- Model:** VMware Int. VMware Virtual Platform
- Processor Type:** Intel(R) Xeon(R) CPU E5540 @ 2.00GHz (highlighted in red)
- Logical Processors:** 2
- VCpus:** 3
- Virtual Machines:** 2
- State:** Up
- Last Update:** 1 day
- Memory:** USED: 87.00 MiB, FREE: 4.75 GB
- Storage:** USED: 28.37 GB, FREE: 10.35 GB, CAPACITY: 39.94 GB

A yellow message box at the bottom left states: "ESXi Shell for the host has been enabled." A green 'Success' message box on the right says: "vMotion is now supported on this host." On the right side, the 'Recent Tasks' pane lists several completed tasks, and the 'Work in Progress' pane is empty.

I then switched to the 'Manage -> Networking' tab and checked to see that a vSphere vMotion port group existed for vSwitch2 as well as that the port was a VMkernel port and make sure the physical adapter has been set correctly ,defined as 1000 and full.



I did the same thing and entered the same configuration on my second ESXi host.



I then navigated to 'Home -> vCenter -> Storage', selected the 'SharedDatastore' datastore, navigated to the 'Manage -> Settings -> Connectivity and Multipathing' tab, and ensured that both of my ESXi hosts were displayed in the table.

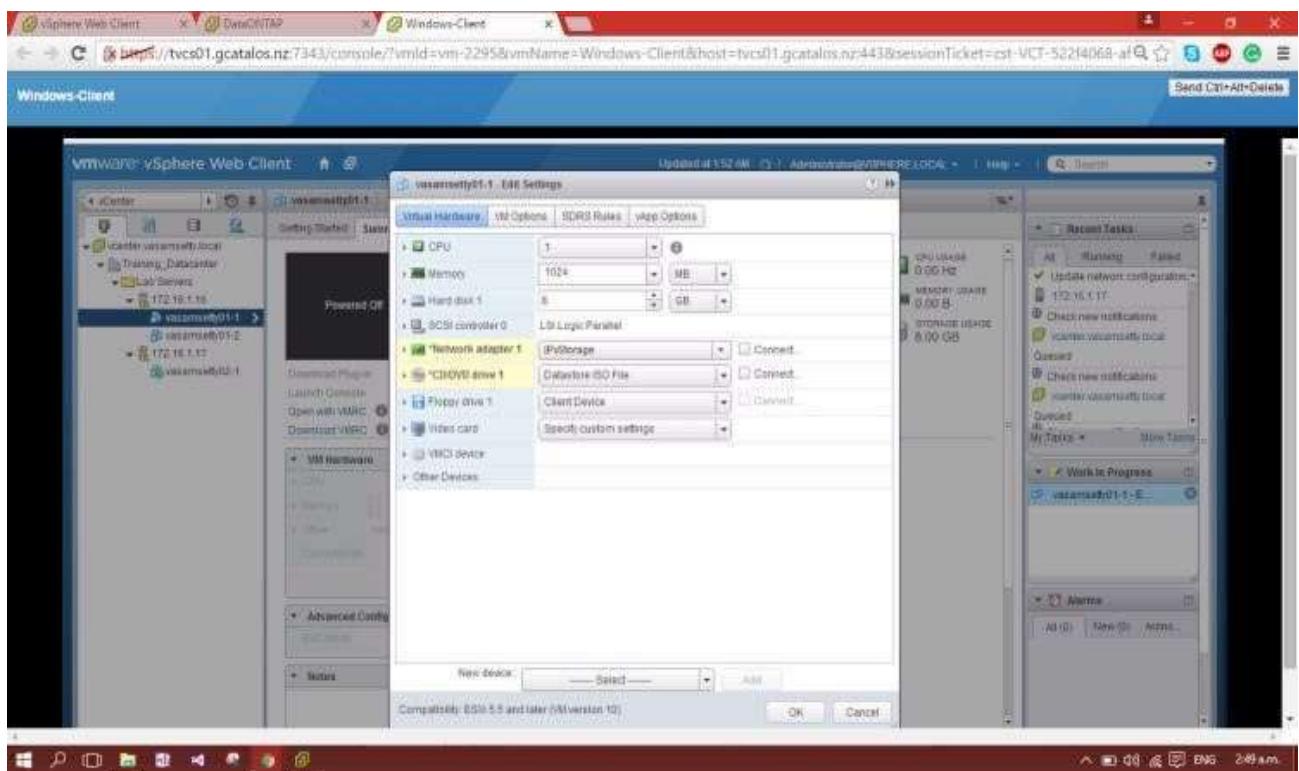
The screenshot shows the VMware vSphere Web Client interface. On the left, the navigation tree shows 'vcenter.vasamsetty.local' with 'Shared Datastores' selected. In the center, under 'Manage', the 'Connectivity and Multipathing' tab is active. A table lists two hosts: '172.16.1.16' and '172.16.1.17'. The 'Mount Status' column shows 'Mounted' for both, and the 'Delete Connectivity' column shows 'Connected' for both. The right side of the screen displays the 'Scheduled Tasks' and 'Work in Progress' panes.

Host	Mount Status	Delete Connectivity
172.16.1.16	Mounted	Connected
172.16.1.17	Mounted	Connected

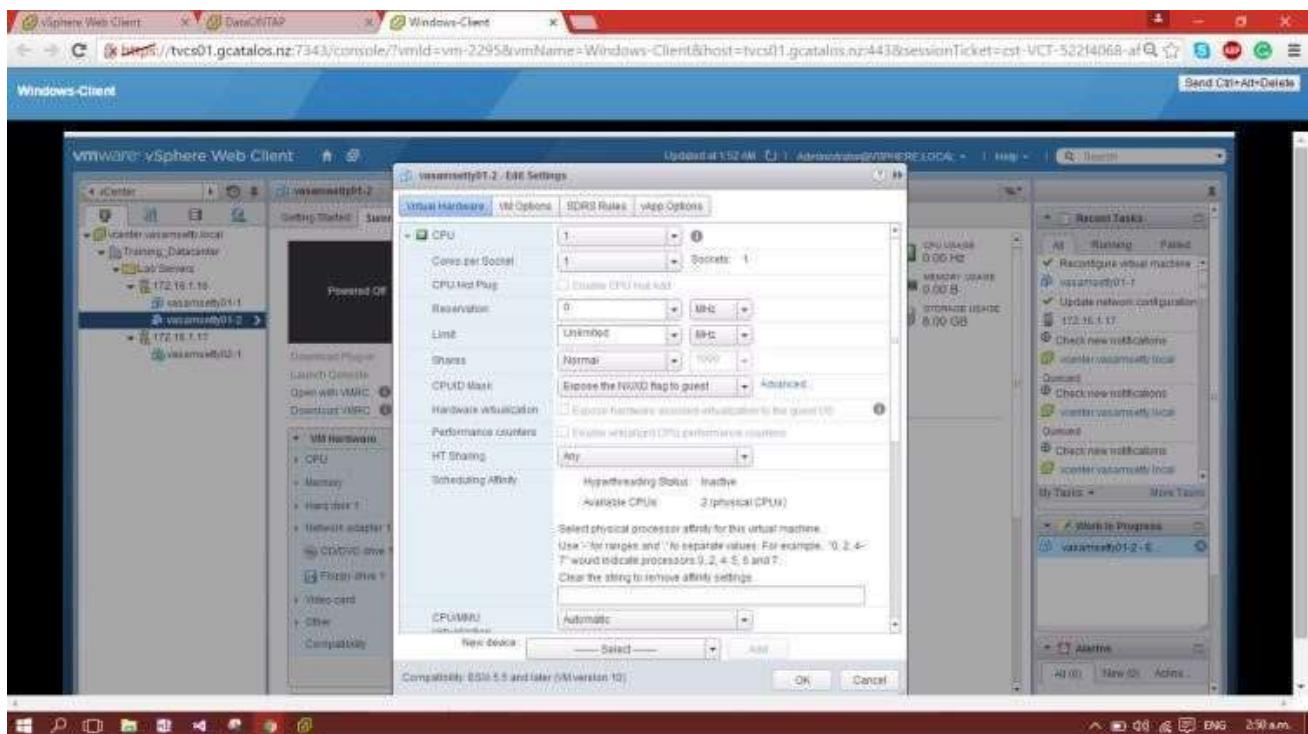
### Verify that your Virtual Machines meet the vSphere vmotion requirements

According to my lab requirements i navigated to my

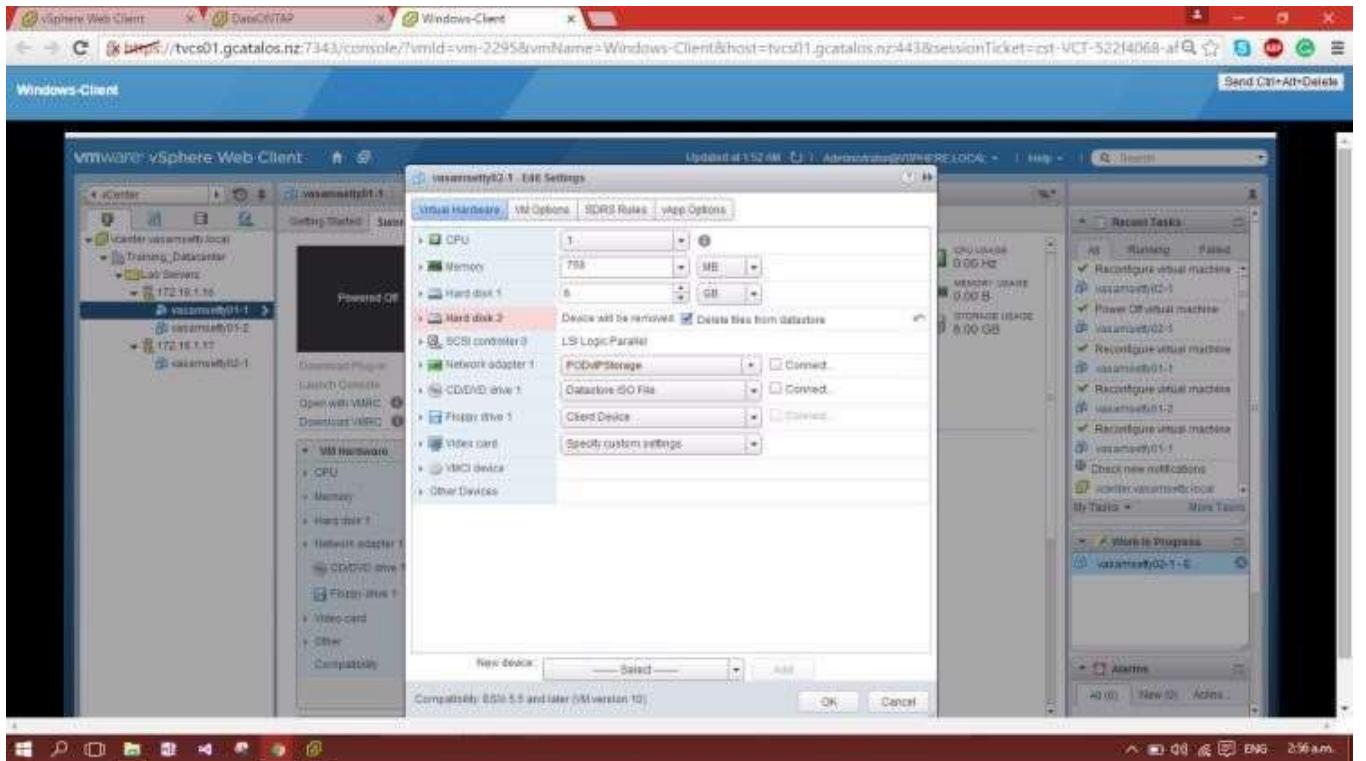
Vcenter>Hosts and Clusters>vasamsetty01-1 VM and right clicked and selected Edit settings. In the Edit settings tab i make sure that i unchecked the CD/DVD drive and Network Adapter 1 as these are the pre-requesties that are need to be done.



As they are no additional hard disk attached to my VM. Next, I verified that the CPU affinity was not set. There was no value under Scheduling Affinity so that was all good.

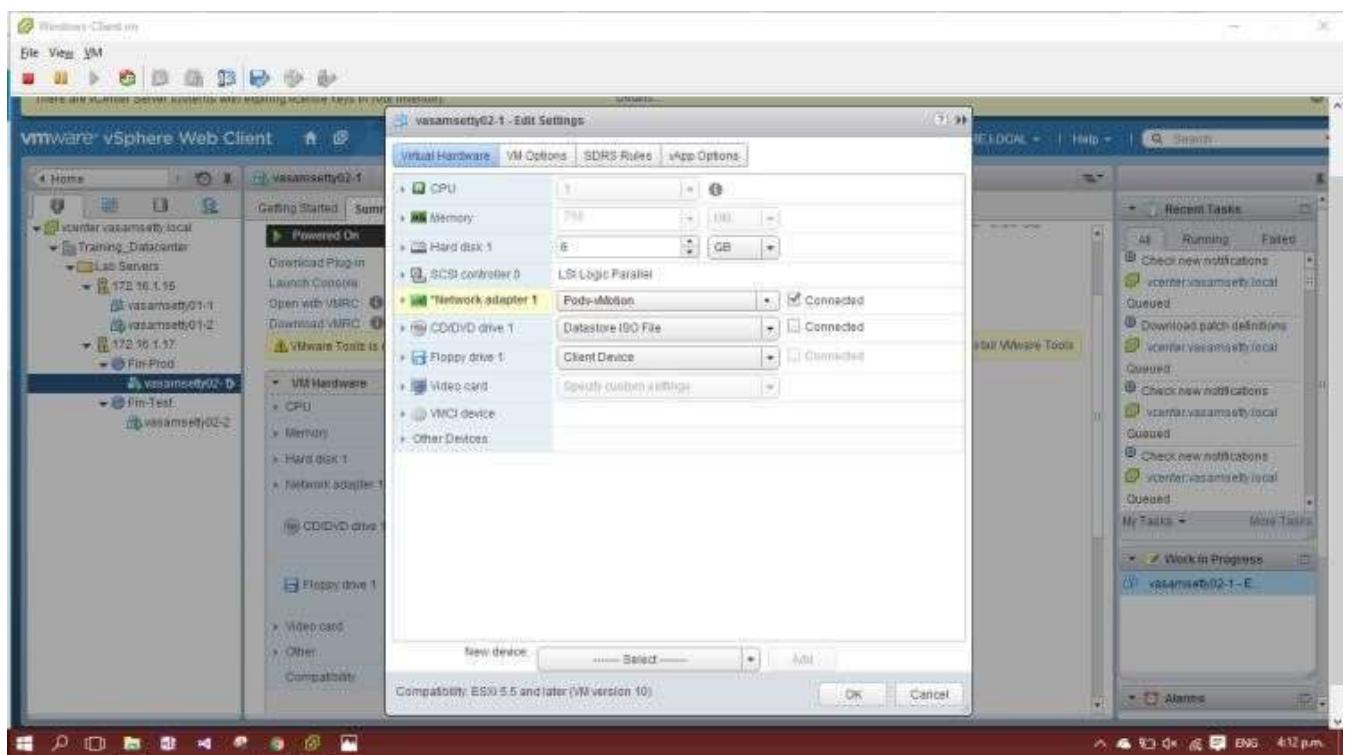


After completing these changes, I repeated the process for my other virtual machines. I also deleted the second hard drive from my 'vasamsetty02-1' virtual machine; the one that was used along with extpart.exe.

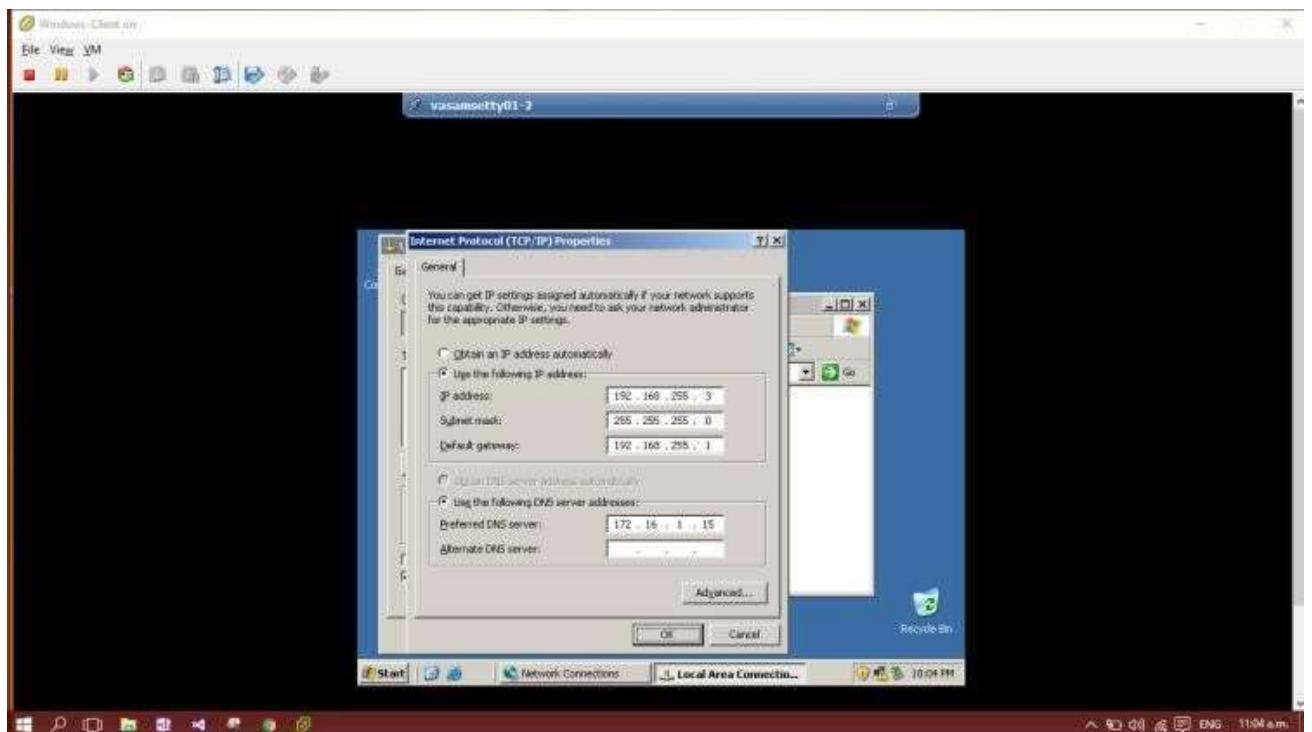


### Performing a vSphere vMotion Migration of a Virtual Machine on a Shared Datastore

In this task i performed all the actions on my second host. From there, i selected my vasamsetty02-1 . To start the process firstly I disconnected my previous portgroup VMNetwork to and connected my new NIC PoDv-vMotion

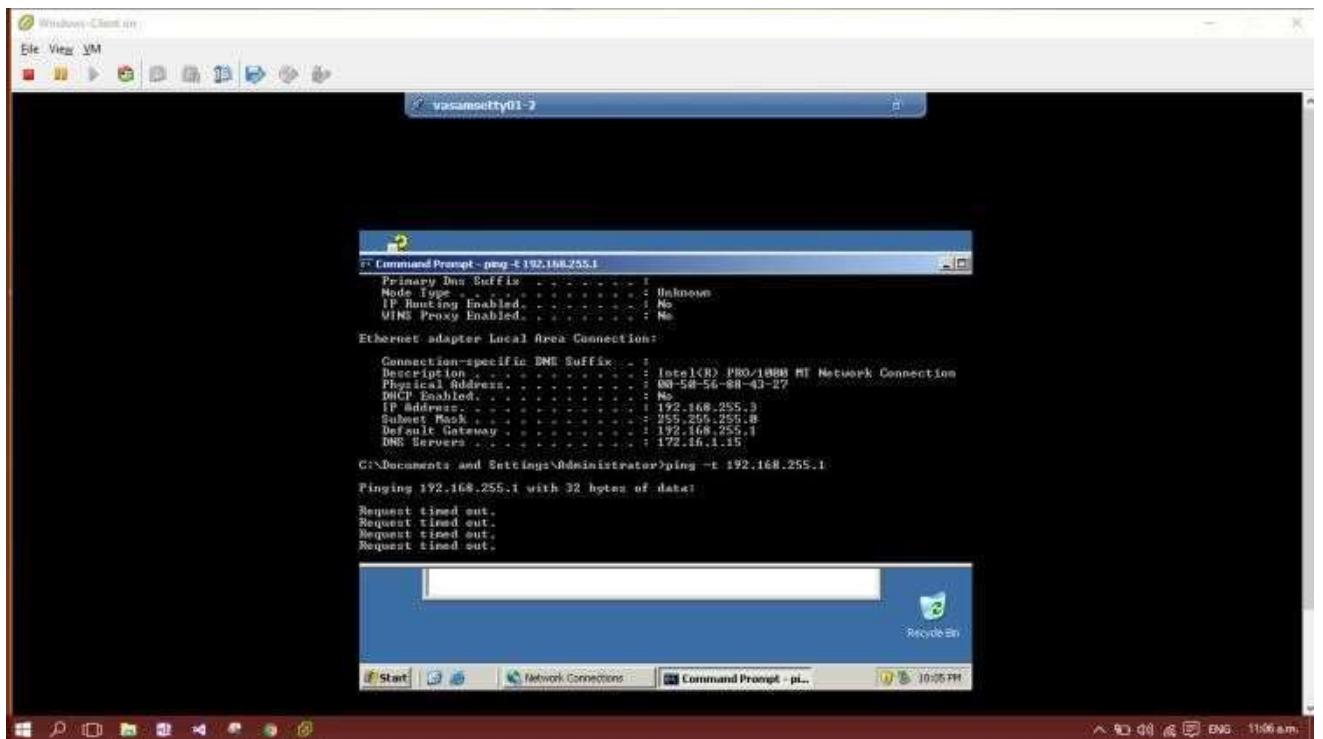


I then needed to change the network settings for the virtual machine, so I changed them as follows (as per my network diagram):

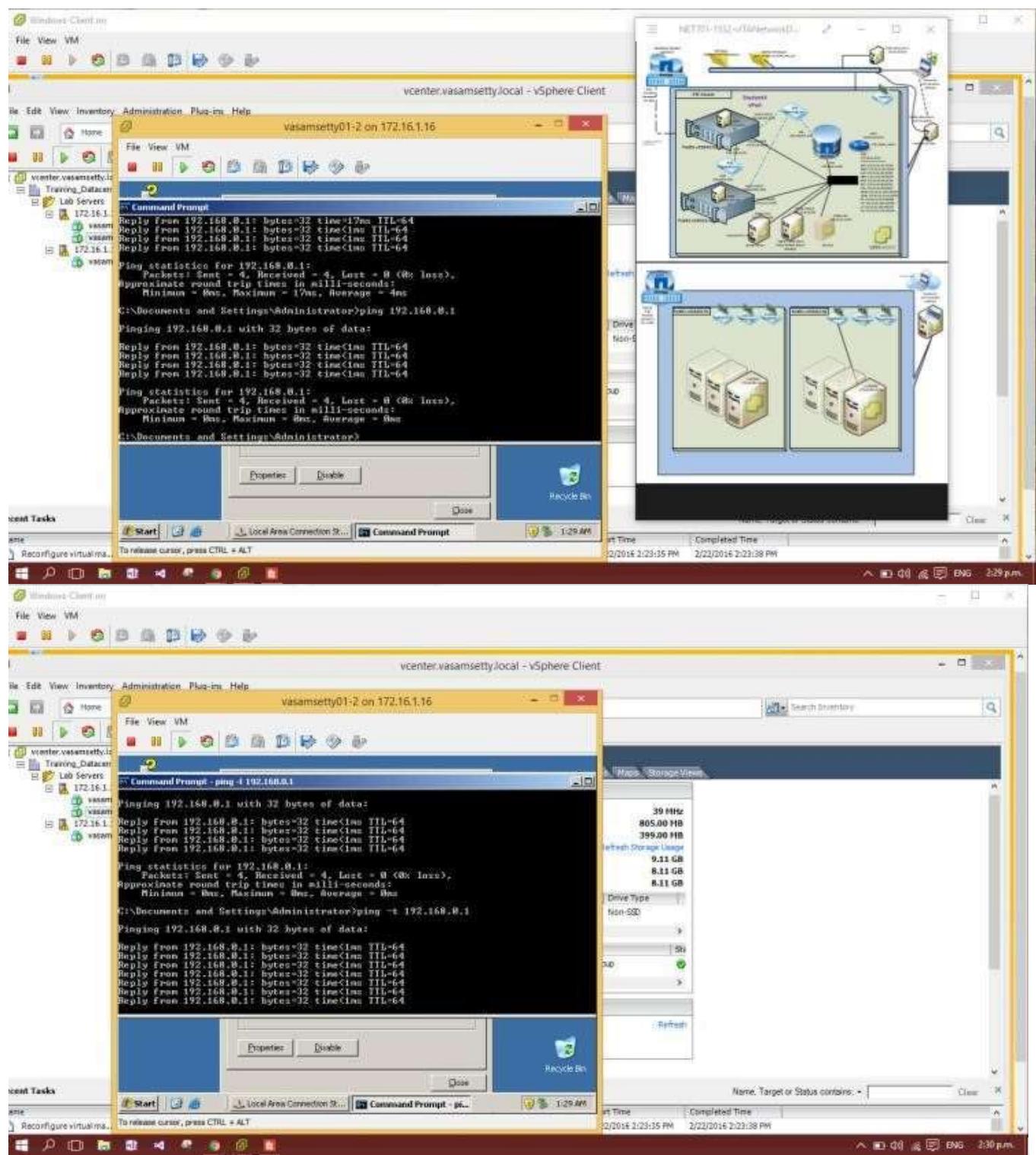


Once my configuration is done. I tried few command prompt functions such as ipconfig, ping after that i entered the command

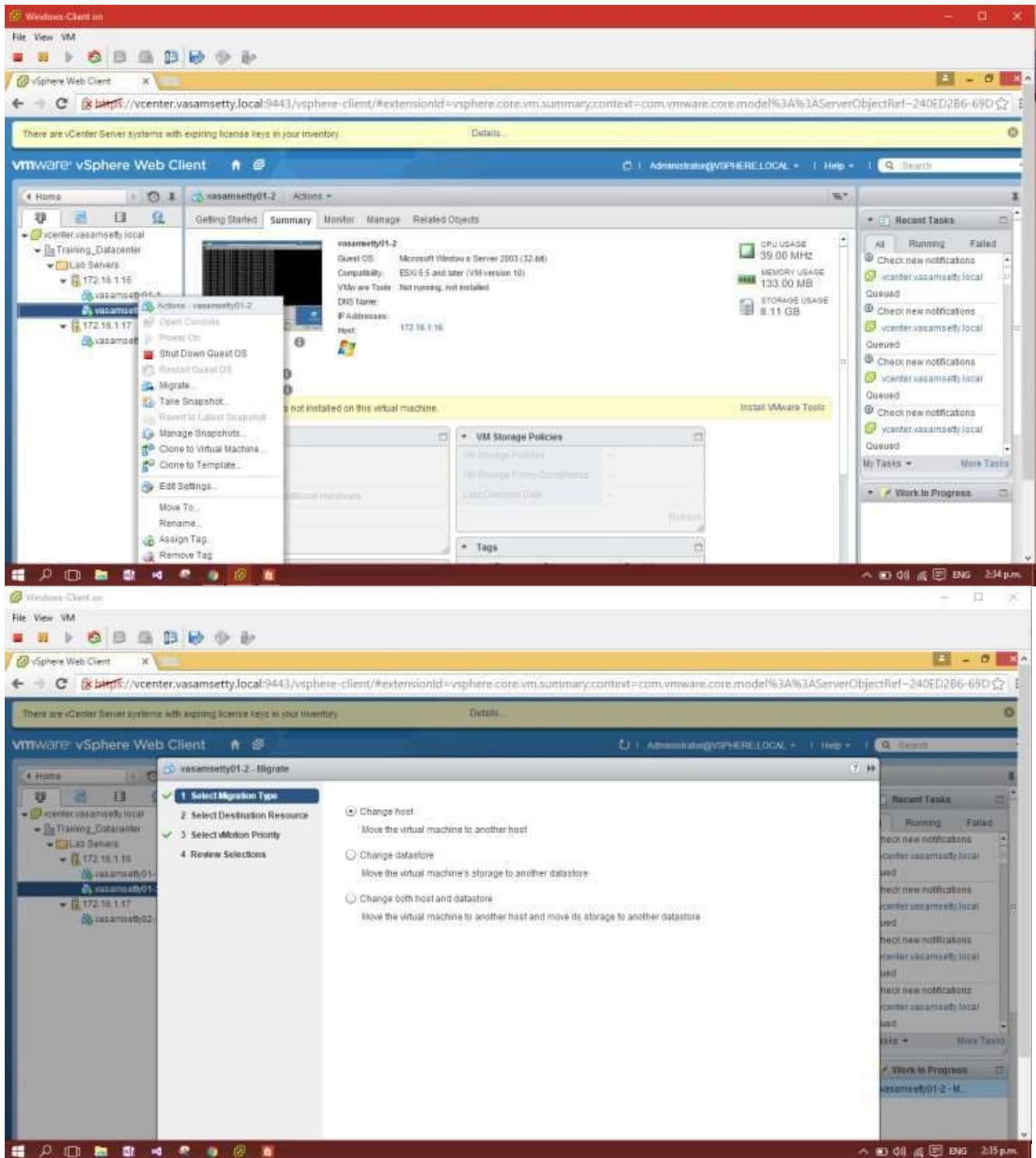
**ping -t 192.168.255.1** in order to get continuously ping the virtual gateway. this gives me an overview and helps to check the dropping of packets during the vMotion.



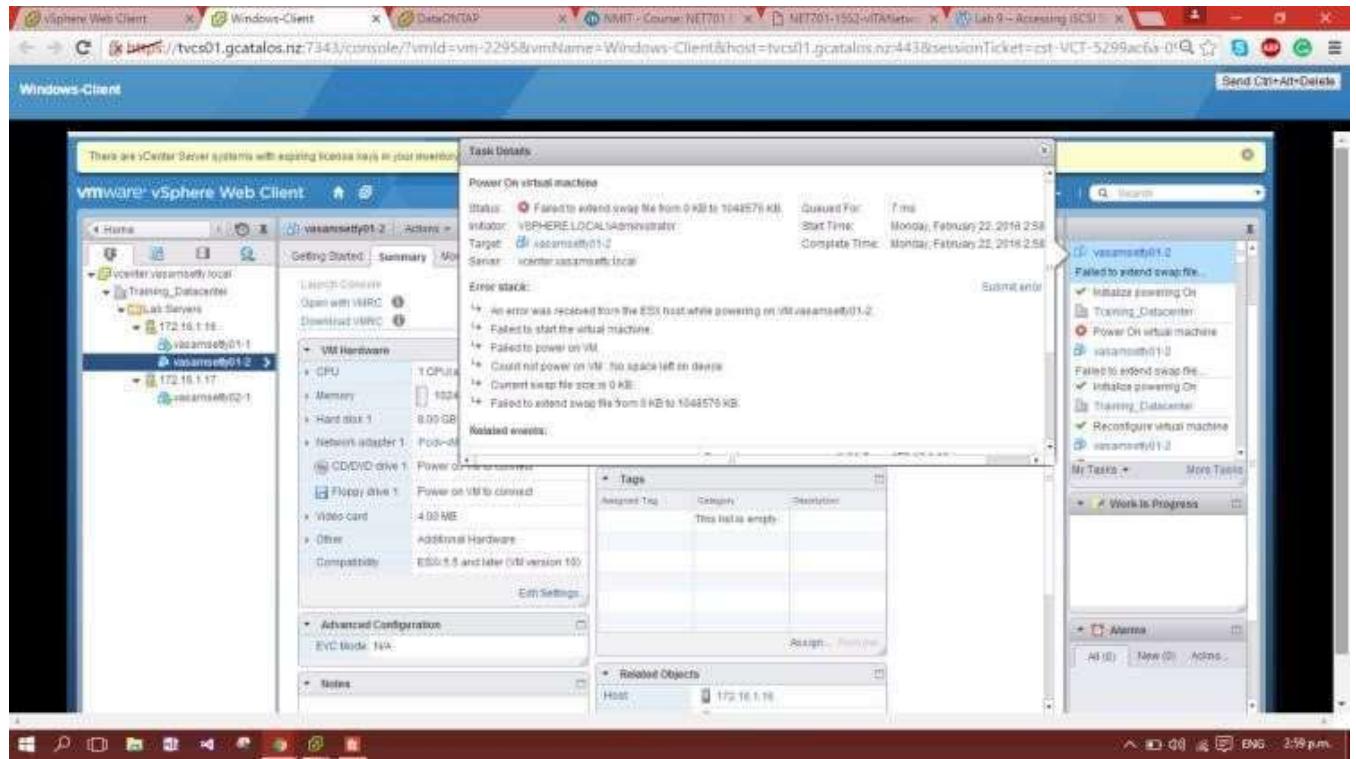
at first i got many errors while pinging, sometimes it showed me no packets has sent and received and after some troubleshoot i got request time out as an error.Finally i figured out what's the problem its because I am confused in providing IP addresses to the virtual switches . This time i make sure to assign the right IP's and finally the ping was successful.



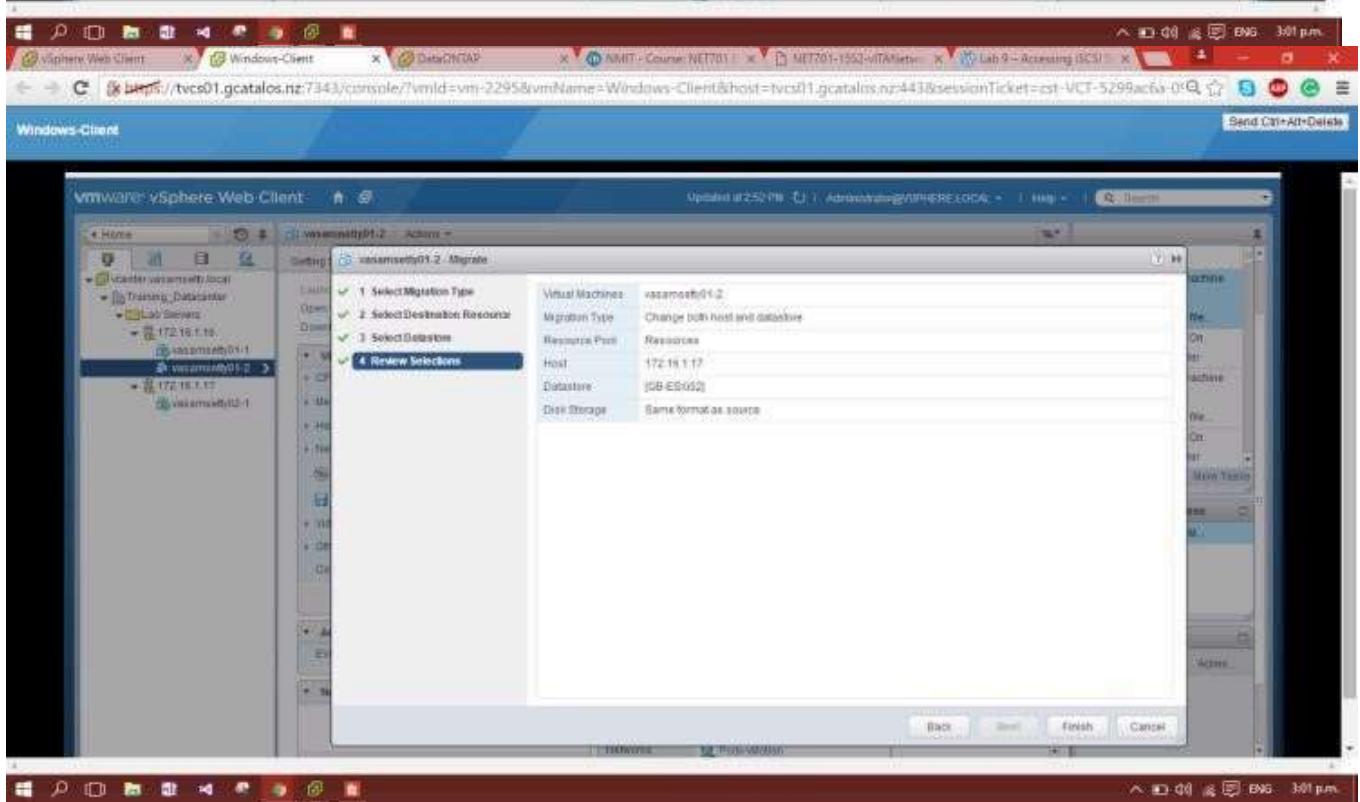
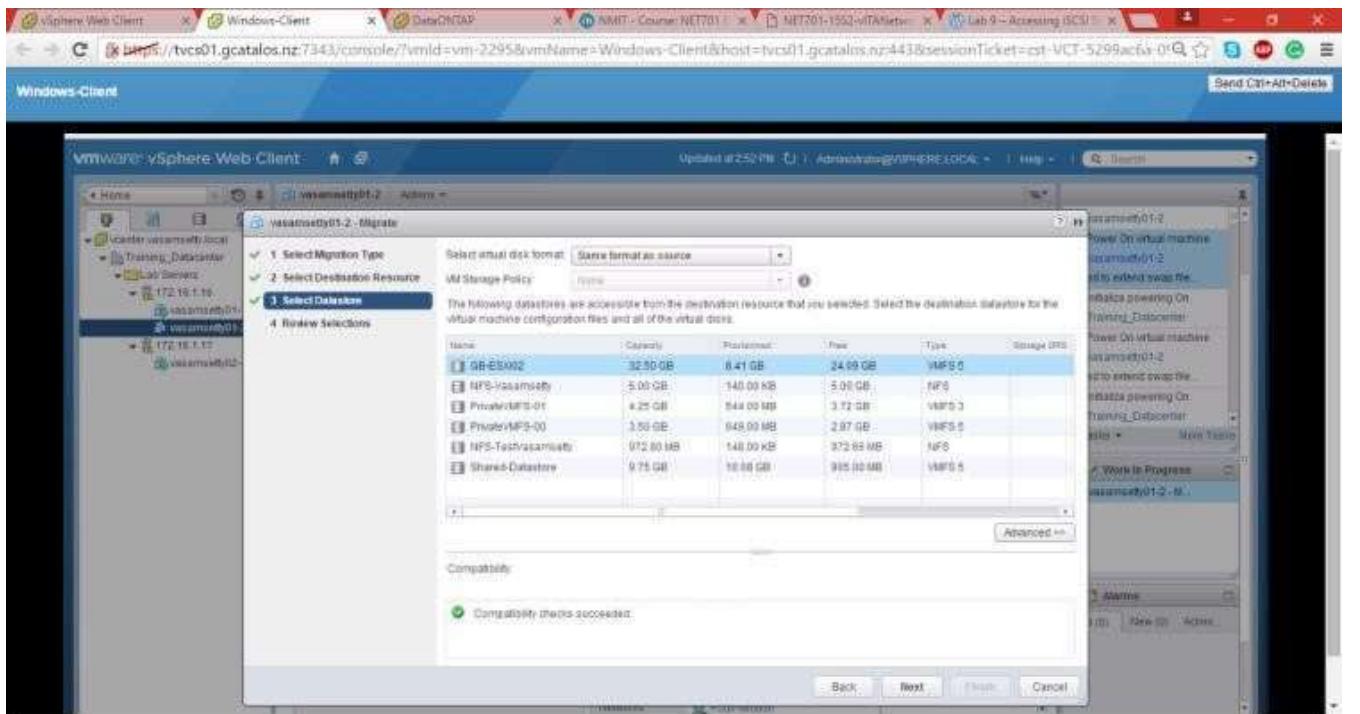
I then returned to the vSphere Web Client, right-clicked the 'vasamsetty01-2' virtual machine, and selected 'Migrate...' from the menu. I was presented with the Migrate Wizard. I selected the Change Host radio button and clicked next.



I got many errors while changing the host because it always show me an error stating that **Failed to extend the Swap File**. This states that my host doesn't have the sufficient memory to make an migration.



So to solve this error i created a shared datastore and make sure that the two host can access the datastore and continued my task instructions and this time no compatibility issues i had selected the shared datastore and taken a review and observe my VM has been migrated successfully



## M.SC. IT Sem 3

## Server Virtualization using VMWare

The screenshot shows the vSphere Web Client interface. On the left, the navigation tree displays the host 'vcenter.varennal.local' under 'Training\_Datacenter' and 'Lat Servers'. A specific virtual machine, 'varamm01-2', is selected and highlighted in blue. The main pane shows the 'Summary' tab for 'varamm01-2'. Key details include:

- Powered On**
- Guest OS:** Microsoft Windows Server 2008 (32-bit)
- Compatibility:** ESXi 5.5 and later (VM version 10)
- VMware Tools:** Not installed
- CNS Name:** **IP Address:** 172.16.1.17
- Port:** 172.16.1.17

Resource usage statistics are shown on the right:

- CPU USAGE:** 0.00 %
- MEMORY USAGE:** 0.00 B
- STORAGE USAGE:** 8.00 GB

The Actions panel on the right lists several tasks:

- ✓ Reconfigure virtual machine
- ✗ varamm01-2
- ✗ Power On virtual machine
- ✗ varamm01-2
- ✗ Failed to extend swap file.
- ✓ Initialize powering On
- ✗ Training\_Datacenter
- ✗ Power On virtual machine
- ✗ varamm01-2
- ✗ Failed to extend swap file.

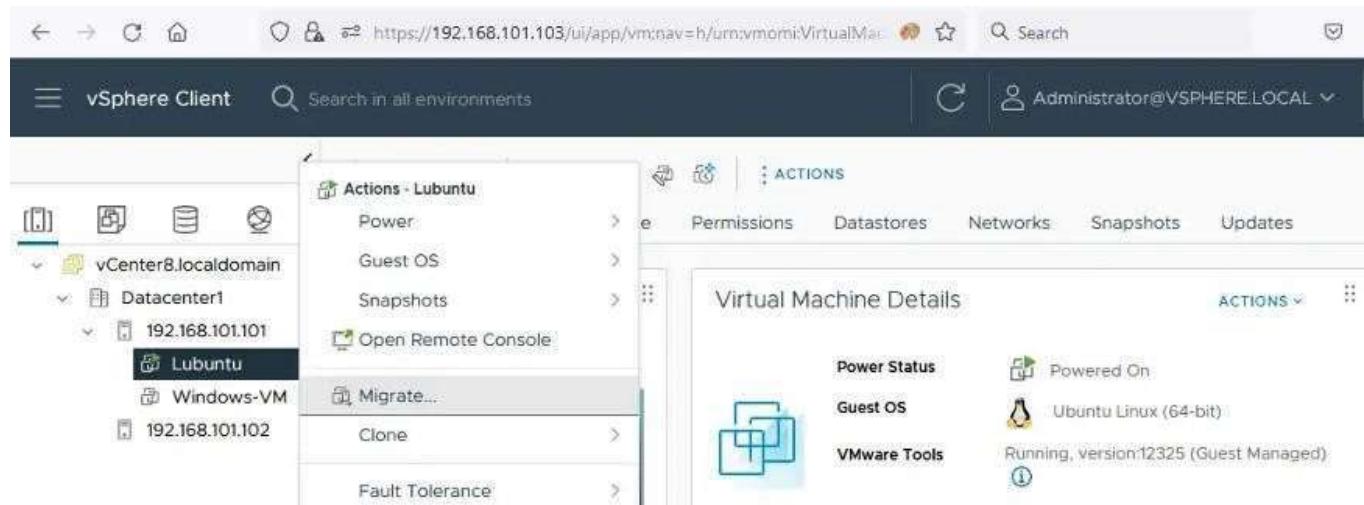
The bottom pane shows the 'Advanced Configuration' section, which is currently collapsed. Other tabs like 'Getting Started', 'Monitor', and 'Manage' are also visible at the top of the main pane.

Perform Compute Resource and Storage Migrations.

The most convenient and effective method to migrate virtual machines to another ESXi host is using vCenter and the vMotion feature. Make sure that both VMs running ESXi hosts and a vCenter VM are operational. Open VMware vSphere Client in your web browser to access your vCenter Server.

VMware provides the vMotion feature for performing VM migration between hosts and datastores. Storage vMotion is used when a VM migrates from one storage to another. vMotion allows you to migrate VMs even if the VMs are running. You can test this great enterprise-level feature in your own ESXi home lab. Try to ping your nested Linux VM during the migration process to check that there is no downtime.

Right-click the VM running on the first ESXi host (ESXi01 – 192.168.101.101) and click **Migrate** in the Actions menu.



The *Migrate* wizard opens:

1. **Select a migration type.** At this step, you can select one of the three options:

- **Change the compute resource only.** Select this option if a VM is located on a shared datastore connected to multiple ESXi hosts.
- **Change storage only.** Select this option if you want to remove the VM from one directly attached datastore to another. Both directly attached datastores are connected to the same ESXi host in this case.
- **Change both compute resource and storage.** Select this option if you want to migrate a VM from one host to another. Thus, both hosts may use different datastores.
- **Cross vCenter Server export.** Use to migrate VMs to a vCenter Server that is a member of another SSO domain.

The third option is suitable in the case reproduced in the current VMware home lab. Both ESXi hosts have their own 40-GB datastores. The running VM will be migrated from one host to another (*ESXi01 > ESXi02*) and from one 40-GB datastore to another (*datastore40 > datastore40-2*).

- Select a compute resource.** This option allows you to define an ESXi host, as well as which CPU and memory resources will be used to run a VM. Select *vCenter8 >*

Migrate | Windows-VM

1 Select a migration type

2 Select a compute resource

3 Select storage

4 Select networks

5 Ready to complete

Select a migration type

Change the virtual machines' compute resource, storage, or both.

- Change compute resource only  
Migrate the virtual machines to another host or cluster.
- Change storage only  
Migrate the virtual machines' storage to a compatible datastore or datastore cluster.
- Change both compute resource and storage  
Migrate the virtual machines to a specific host or cluster and their storage to a specific datastore or datastore cluster.
- Cross vCenter Server export  
Migrate the virtual machines to a vCenter Server not linked to the current SSO domain.

*Datacenter01 > 192.168.101.102* (the IP address of *ESXi02*, which is the second host in this VMware test environment).

Migrate | Windows-VM

1 Select a migration type

2 Select a compute resource

3 Select storage

4 Select networks

5 Ready to complete

Select a compute resource

Select a cluster, host, vApp or resource pool to run the virtual machines.

VM origin @

vCenter8.localdomain

- 192.168.101.101
- 192.168.101.102**

Compatibility

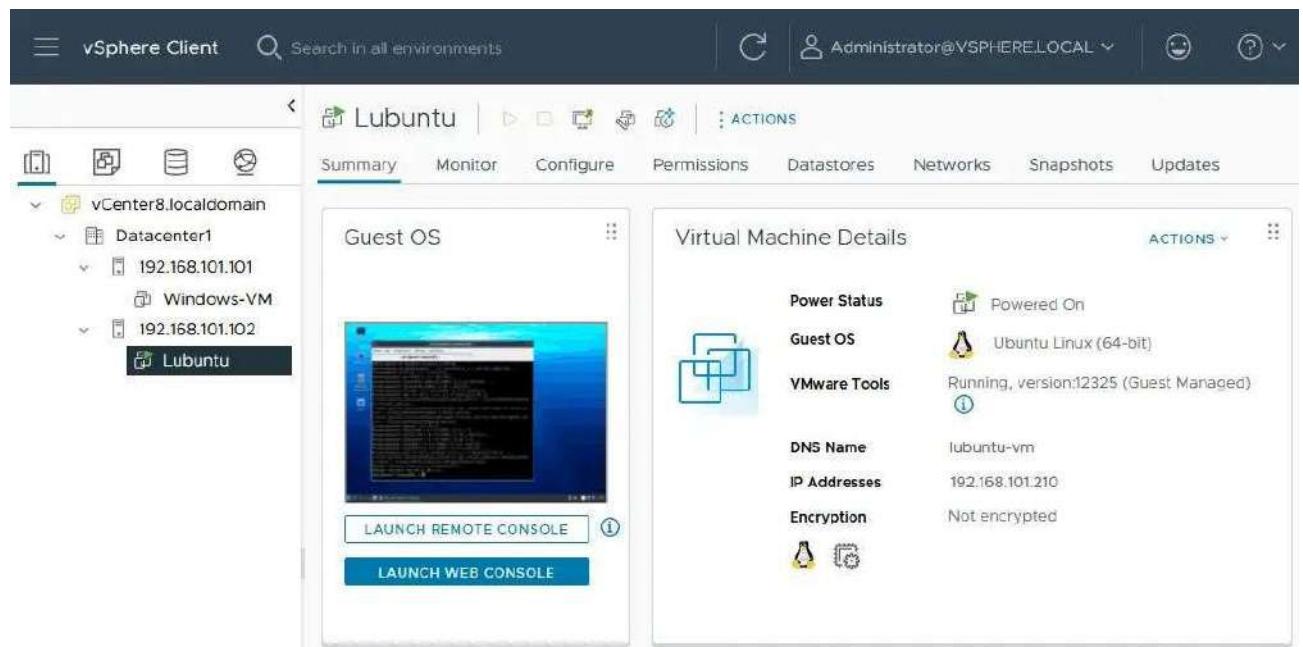
Windows-VM

- 192.168.101.102
- "Device" "CD/DVD drive 1" uses backing "[datastore10a] install/winPreVista.iso", which is not accessible.**

CANCEL BACK NEXT

2. **Select storage.** At this stage, you should select the destination storage for the virtual machine migration. Select the virtual disk format: the same as the source (thin provisioning in this case). Select the datastore (*datastore40-2* in our example).
3. **Select networks.** Select destination networks for the virtual machine migration. *VM Network* is selected for this purpose in our example.
4. **Select vMotion priority.** You can set vMotion to high priority (recommended) or normal priority. Set to high priority.
5. **Ready to complete.** Check your VM migration parameters and hit **Finish** to start the migration.

Wait until the VM migration process is completed. You can open the *Recent Tasks* bar at the bottom of the VMware vSphere Client web interface to check the task's progress. After the VM migration task is completed successfully, your VM will begin running on the second ESXi host (see the screenshot below).

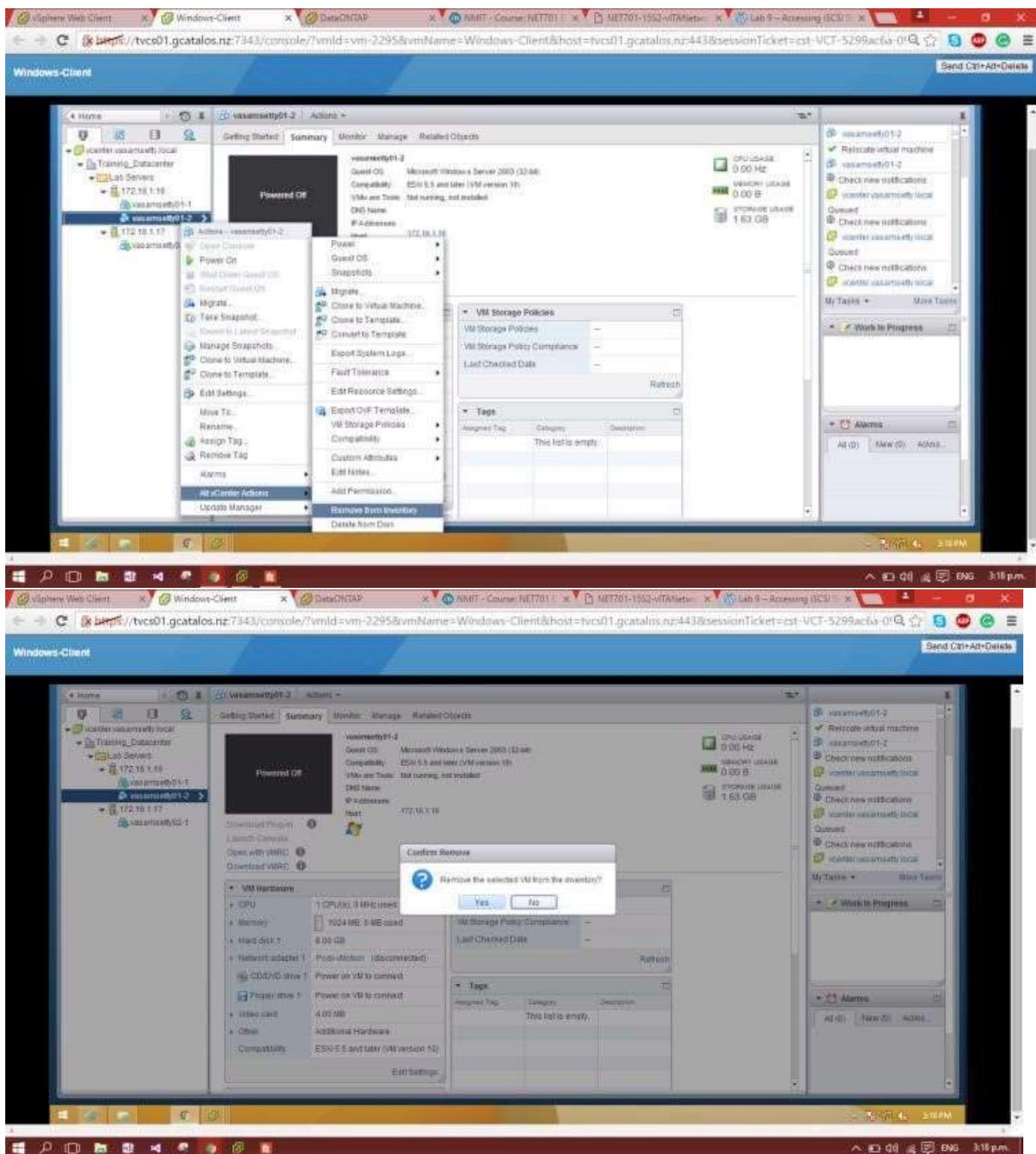


You can see a preview of the Lubuntu VM desktop where the console was opened in order to check the network and install VMware Tools before migration. The virtual machine's running state is preserved.

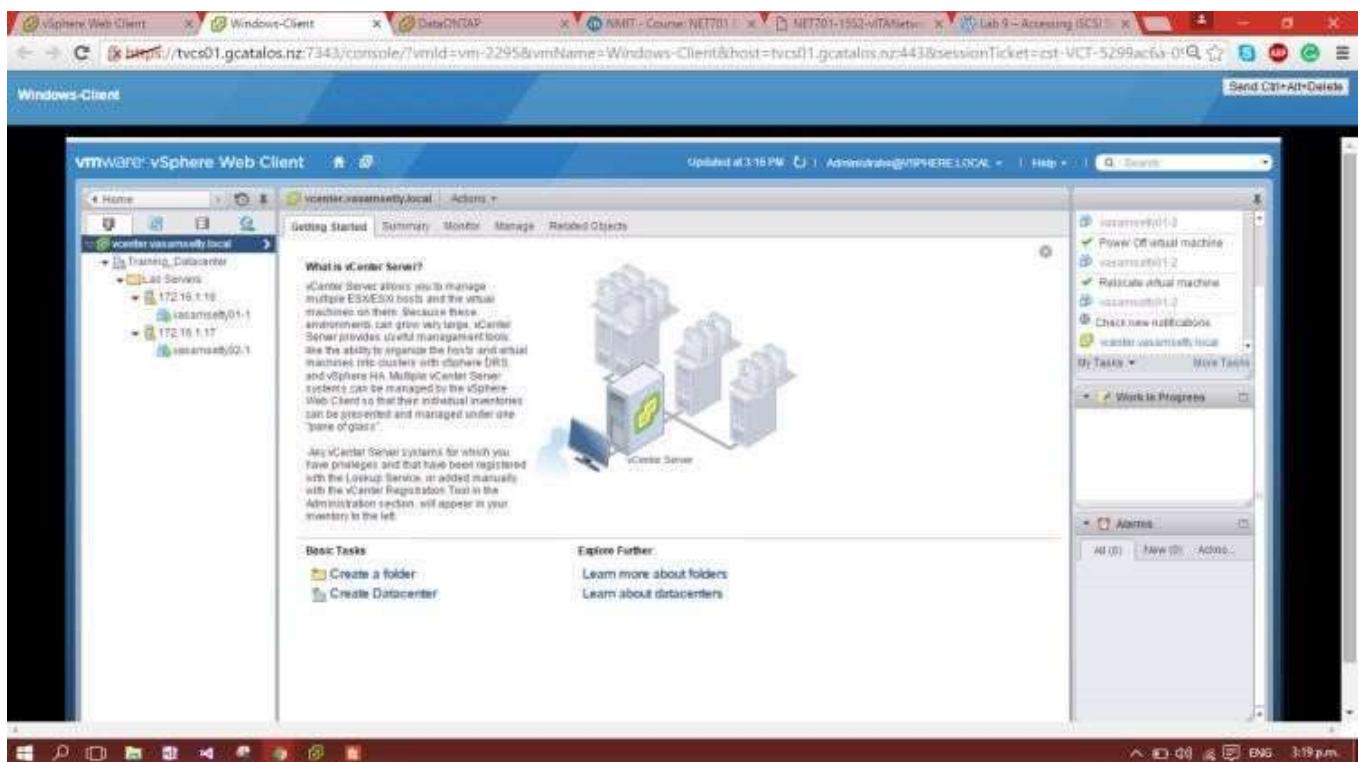
## **Practical No 11 :- Managing Virtual Machines**

### **Unregistering a Virtual Machine in the vCenter Server Inventory**

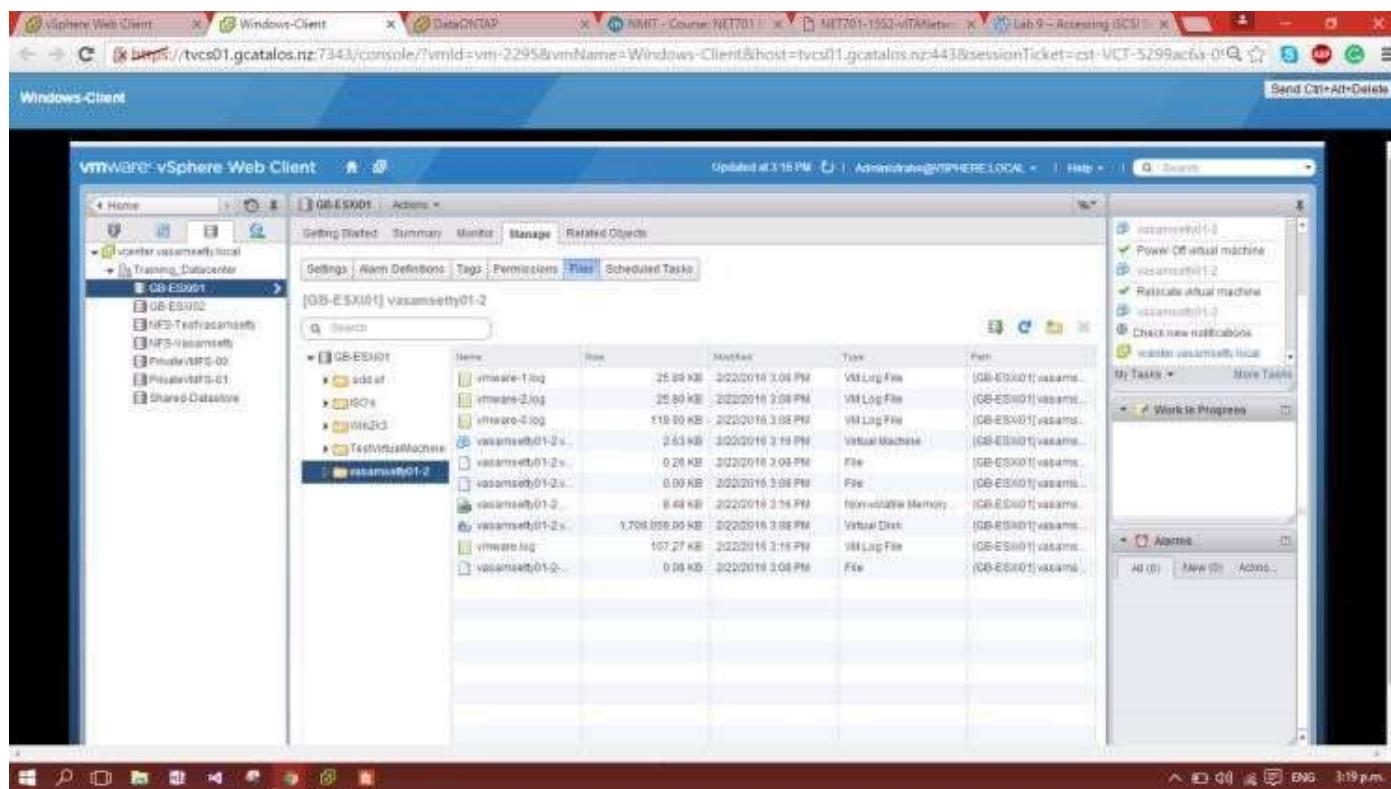
For this task am gonna use my vasamsetty01-2 virtual machine. I right clicked on this virtual machine clicked on All Vcenter Actions and selected Remove from Inventory options. Before doing this i noted down on which datastore my virtual machine exist. My vasamsetty01-2 is in GB-ESXi01 datastore.



I clicked yes and seen that my virtual machine is removed from my existing host.

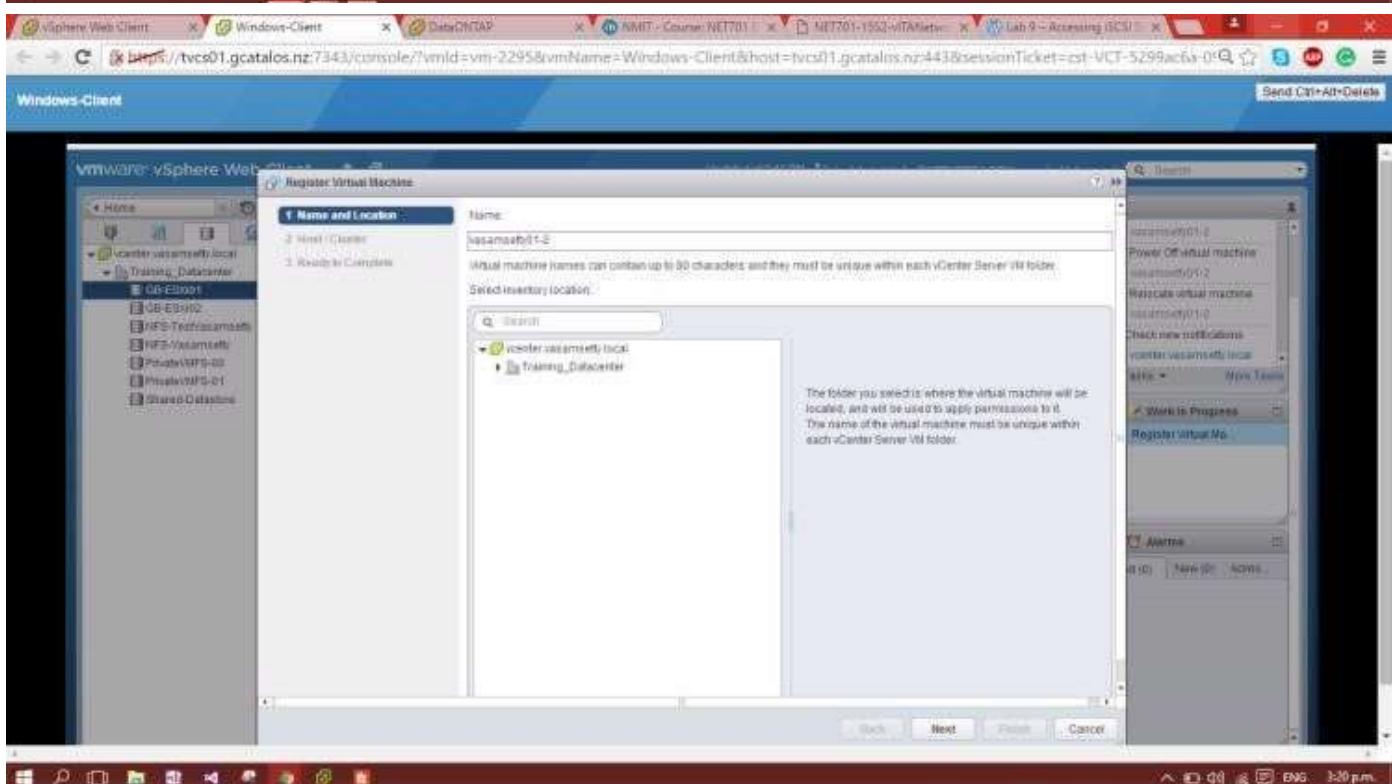
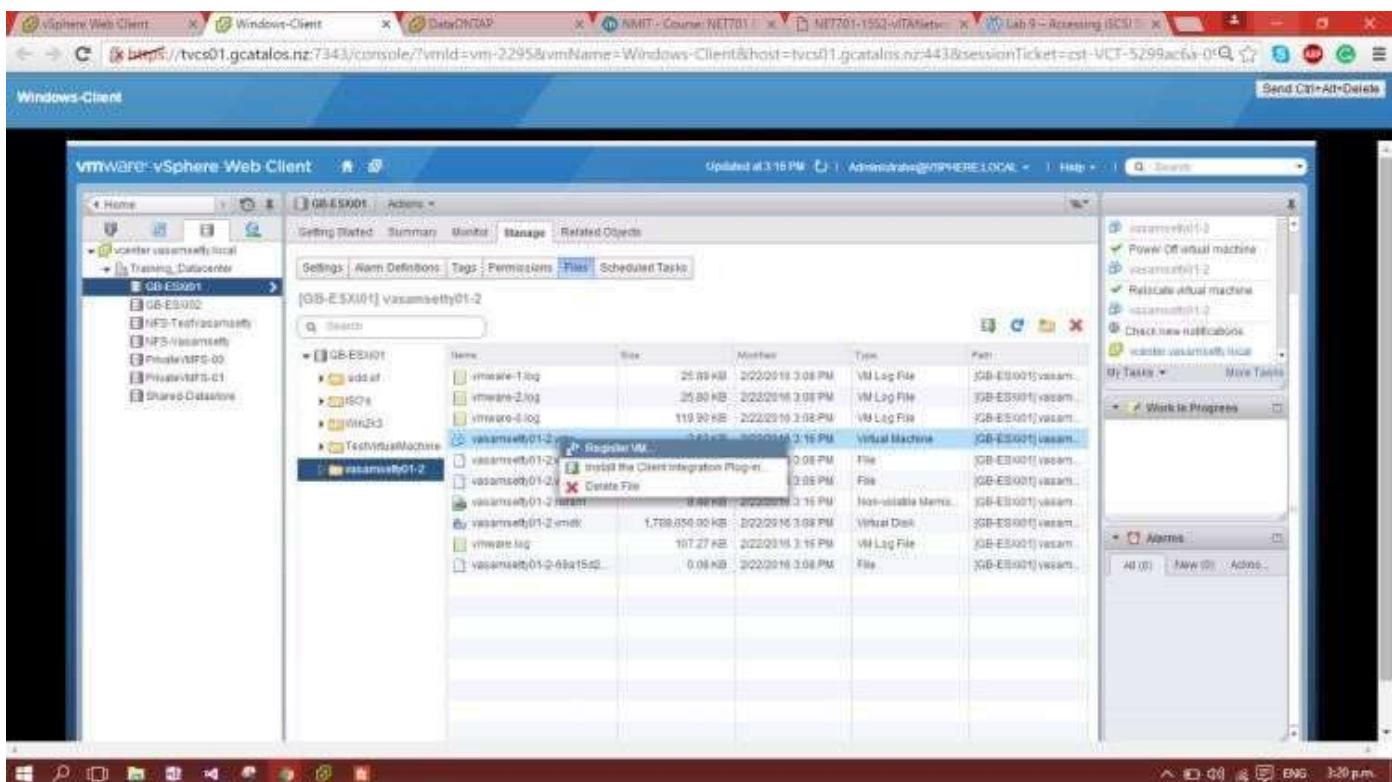


I then navigated to 'Home -> vCenter -> Datastores', clicked on my 'GB-ESXi01' datastore, and clicked on the 'Navigate to the datastore file browser' button. That's a surprise i find vasamsetty01-2 still available in the datastore this shows me that if we click on remove from inventory option it just removes from the host items. the VM files still hold it files in it datastore.



### Registering a Virtual Machine in the vCenter Server Inventory

Now i tried to register again the vm to my host items. For this i right clicked on VM and clicked on Register VM. It made me to start the Guest OS installation wizard. I continued the process and finally my VM is back again.



The screenshot shows the VMware vSphere Web Client interface. The main window displays the 'Register Virtual Machine' wizard, currently at step 2: Host / Cluster. The left sidebar shows the vCenter inventory, including Datacenters, Clusters, Hosts, and Datastores. The right sidebar contains various management options like Power On/Off, Relocate, and Check new notifications. A red notification bar at the bottom right indicates that a screenshot was saved to OneDrive.

**Screenshot saved**  
The screenshot was added to your OneDrive.

**Windows-Client**

**VMware vSphere Web Client**

**Register Virtual Machine**

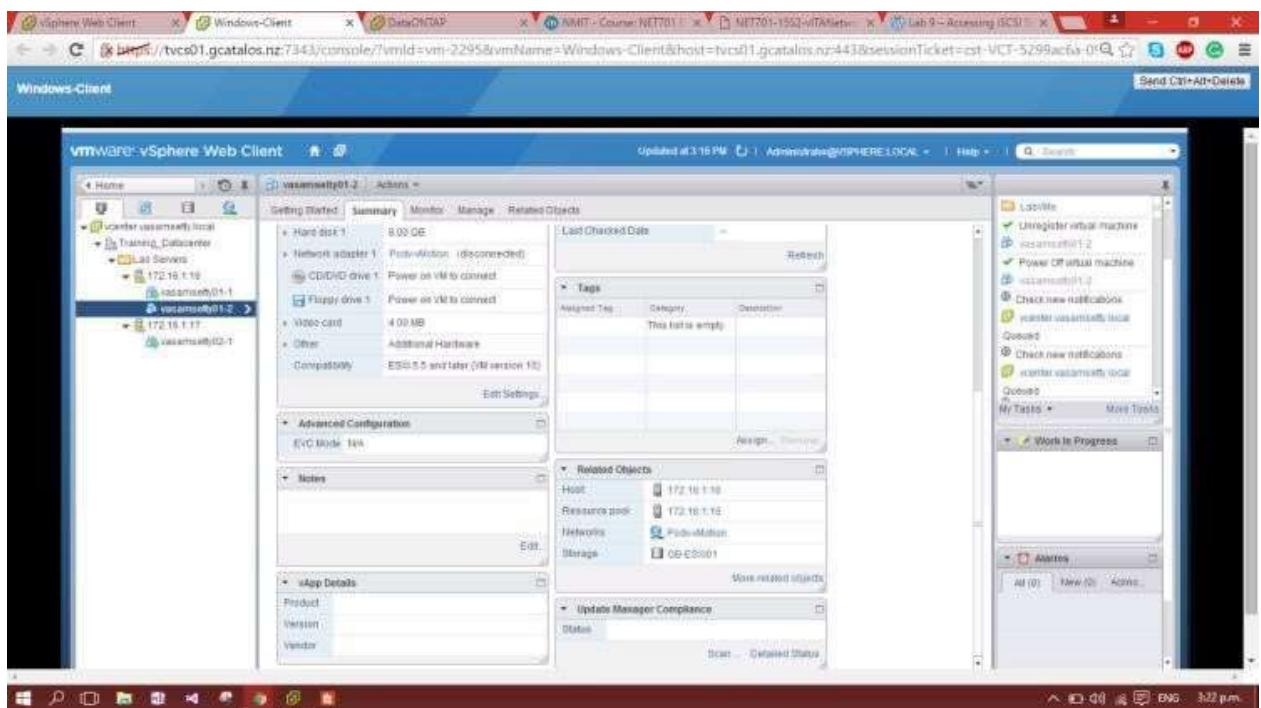
**2 Host / Cluster**

VM Name: vasanm01-2  
Folder Name: LabVM  
Host Name: 172.16.1.16

Training\_Datacenter  
172.16.1.16  
172.16.1.17

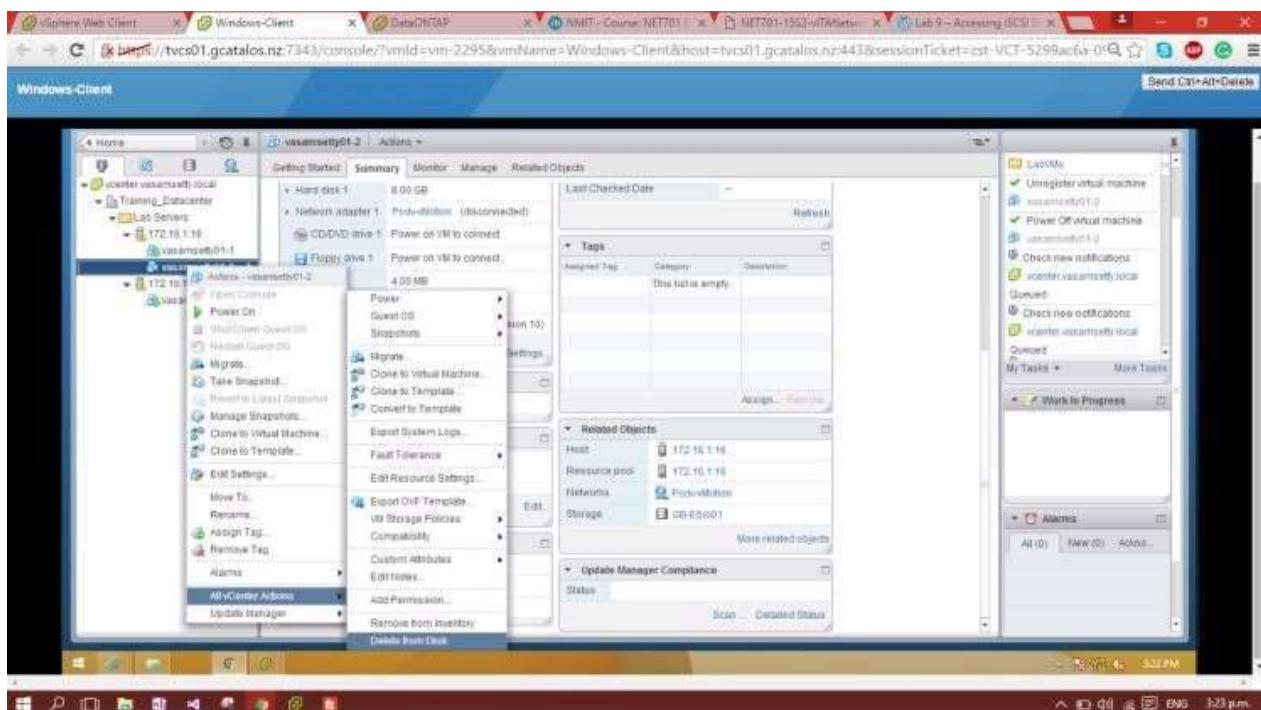
Selected the cluster or host on which to run this virtual machine.

Back Next Finish Cancel

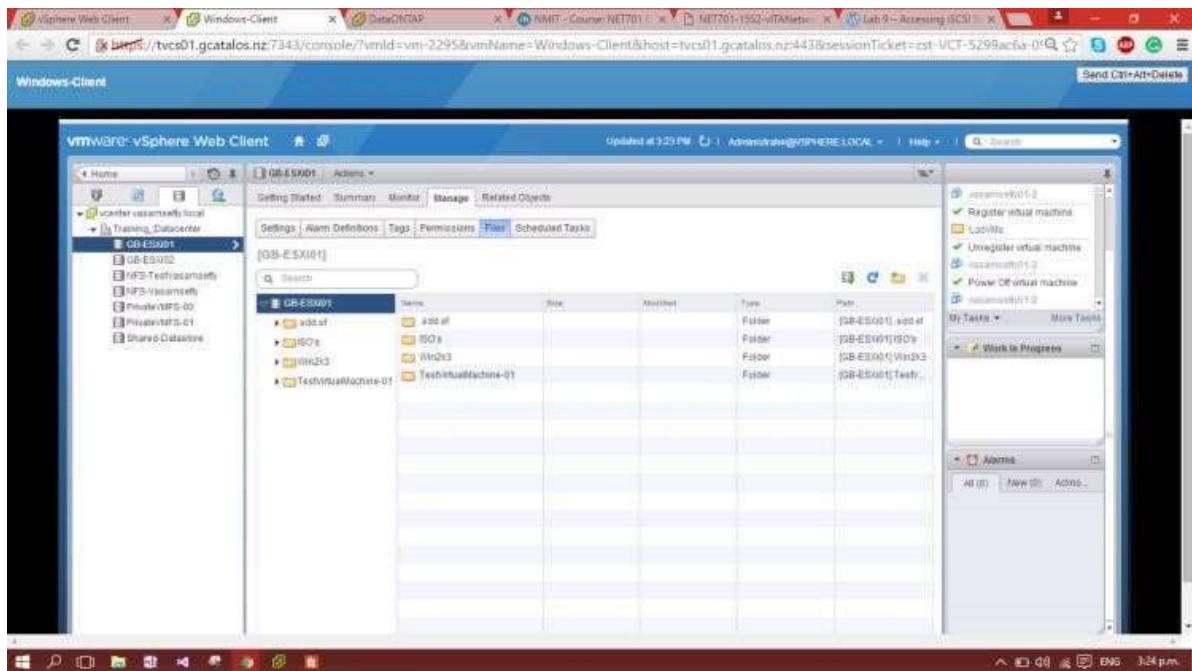


### Unregistering and Deleting Virtual Machines from Disk

To start off this task, I selected the newly created ‘vasamsetty01-2’ virtual machine, clicked the ‘Summary’ tab, and reviewed the Related Objects settings to see what datastore the virtual machine resided on. As you may have suspected, it was on the ‘GBESXI01’ datastore. This time rather selecting Remove from Inventory i choose Delete from Disk.

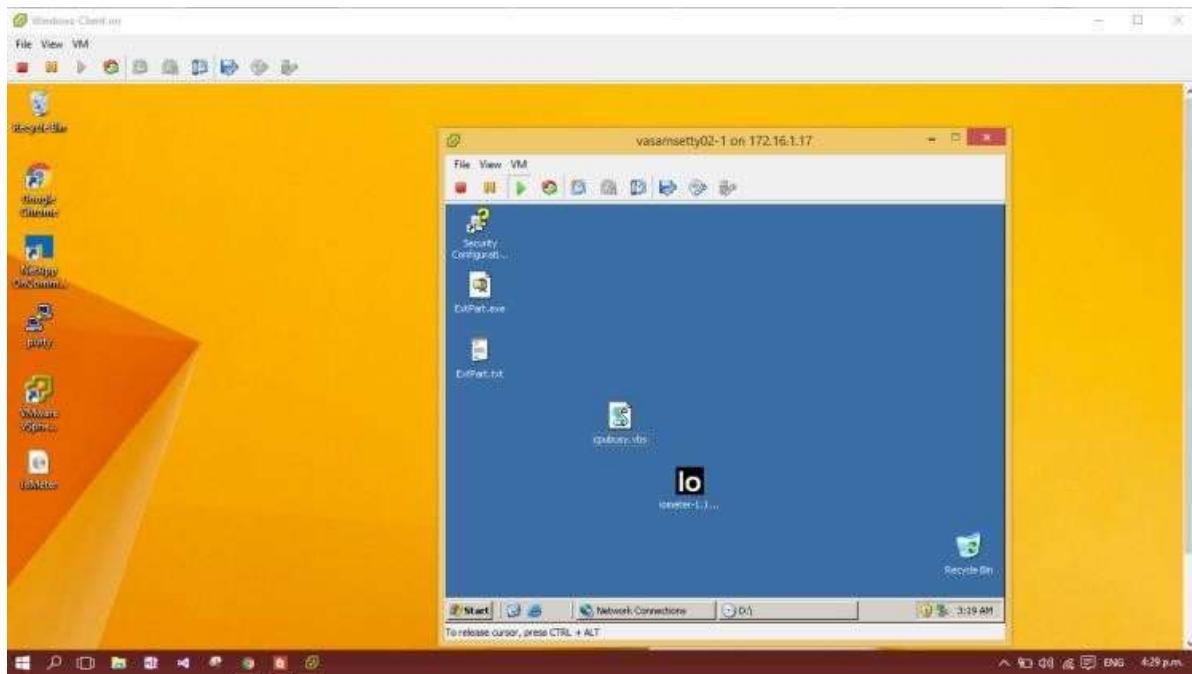


I then navigated to 'Home -> vCenter -> Datastores', selected the 'SharedDatastore' datastore on which the virtual machine resided, and verified that it's folders/files no longer existed.

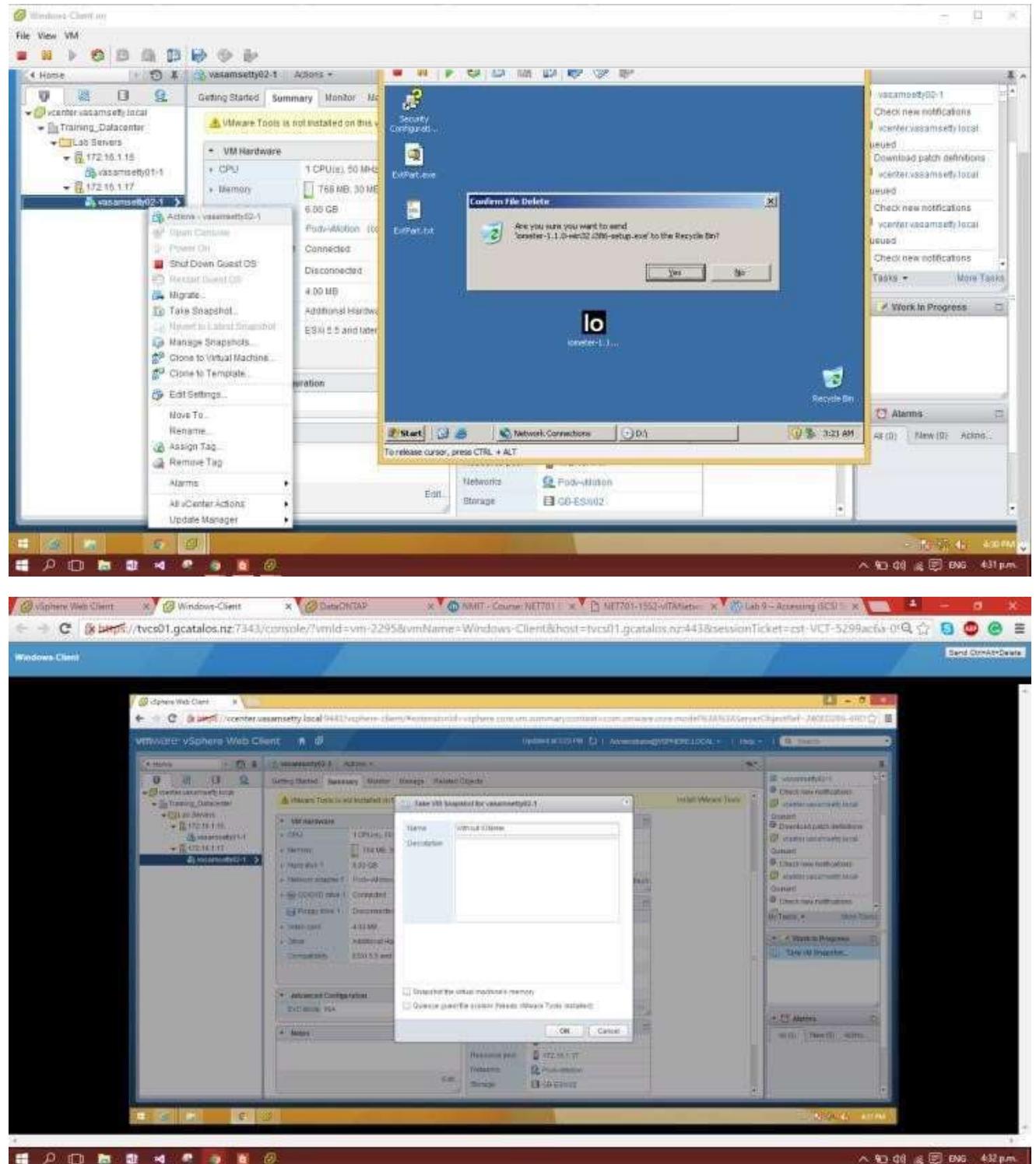


### Taking Snapshots of a Virtual Machine

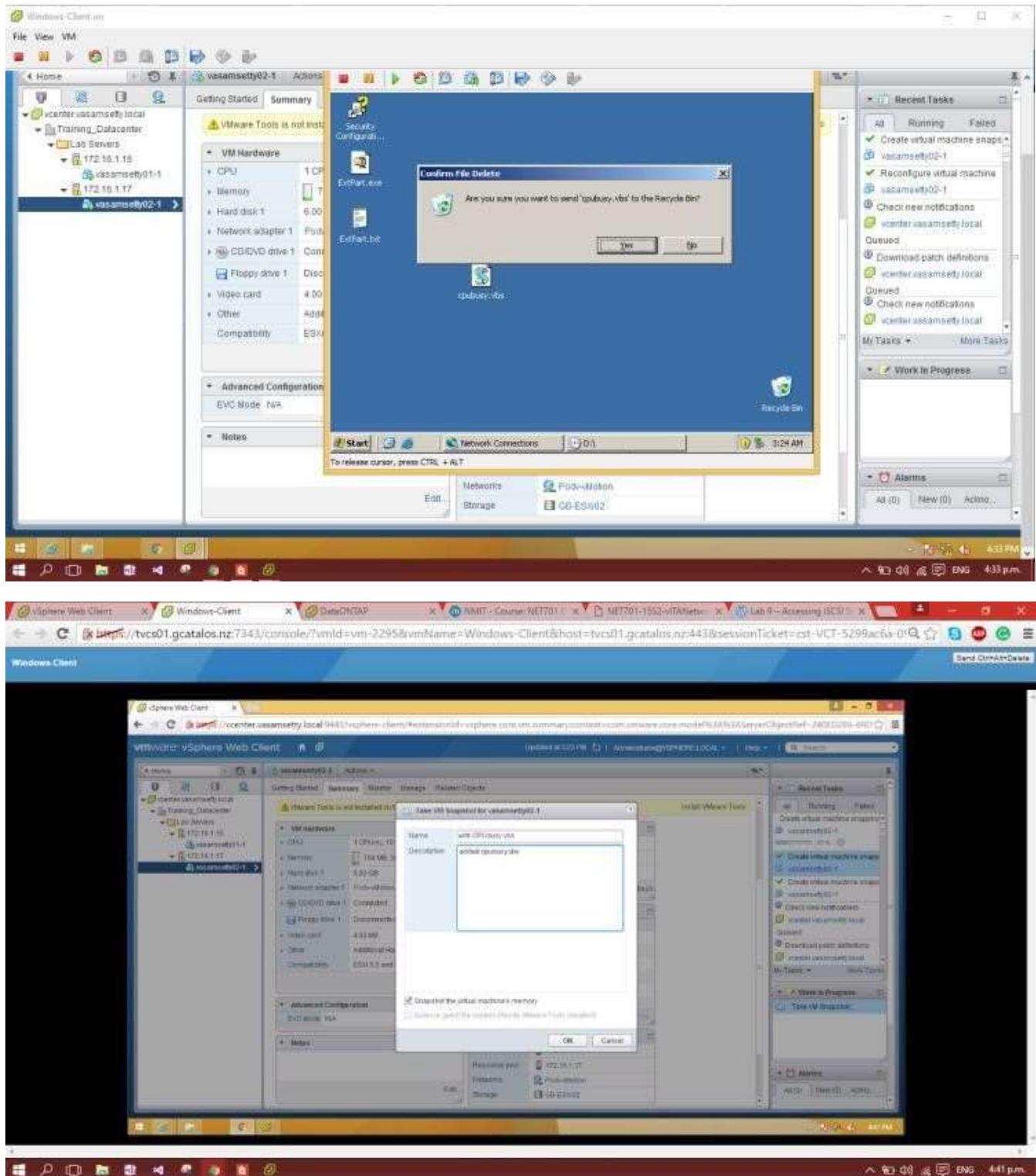
Before doing this task i once again mounted IOMeter iso's to my windows server 2003 and copied the IOMETER.exe and cpibusy.vib file on the desktop.



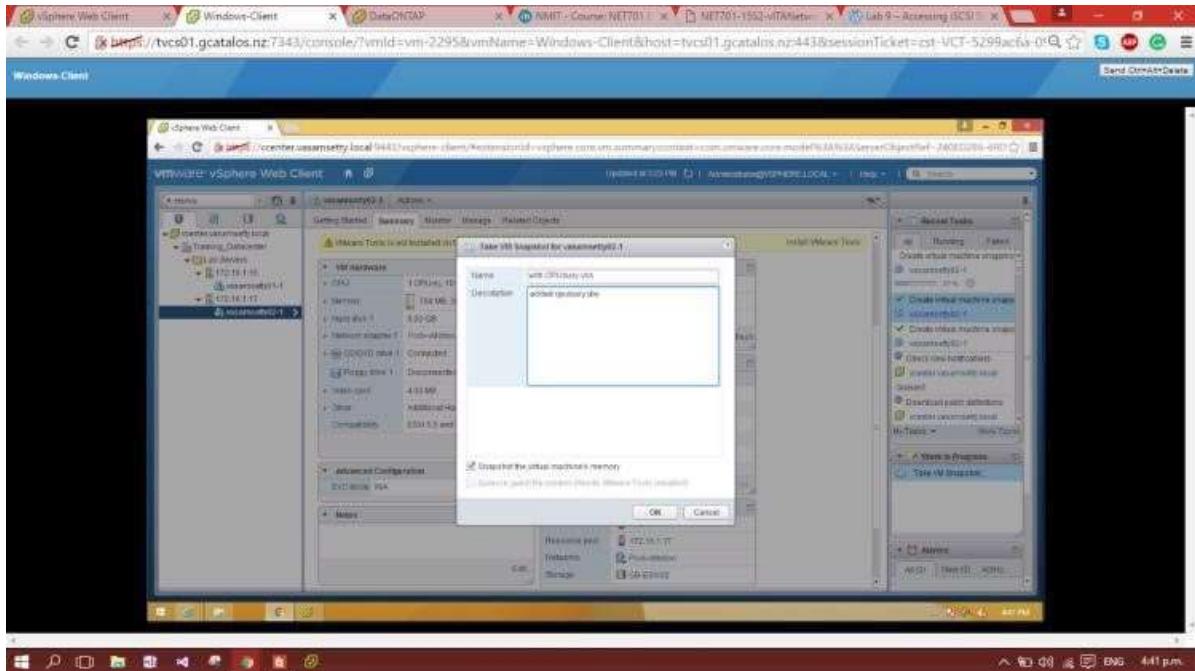
I then deleted iometer as per the lab manual instructions, switched back to the vSphere Web Client, right-clicked my 'vasamsetty02-1' virtual machine, and selected 'Take Snapshot...' from the list. This opened the snapshot wizard.



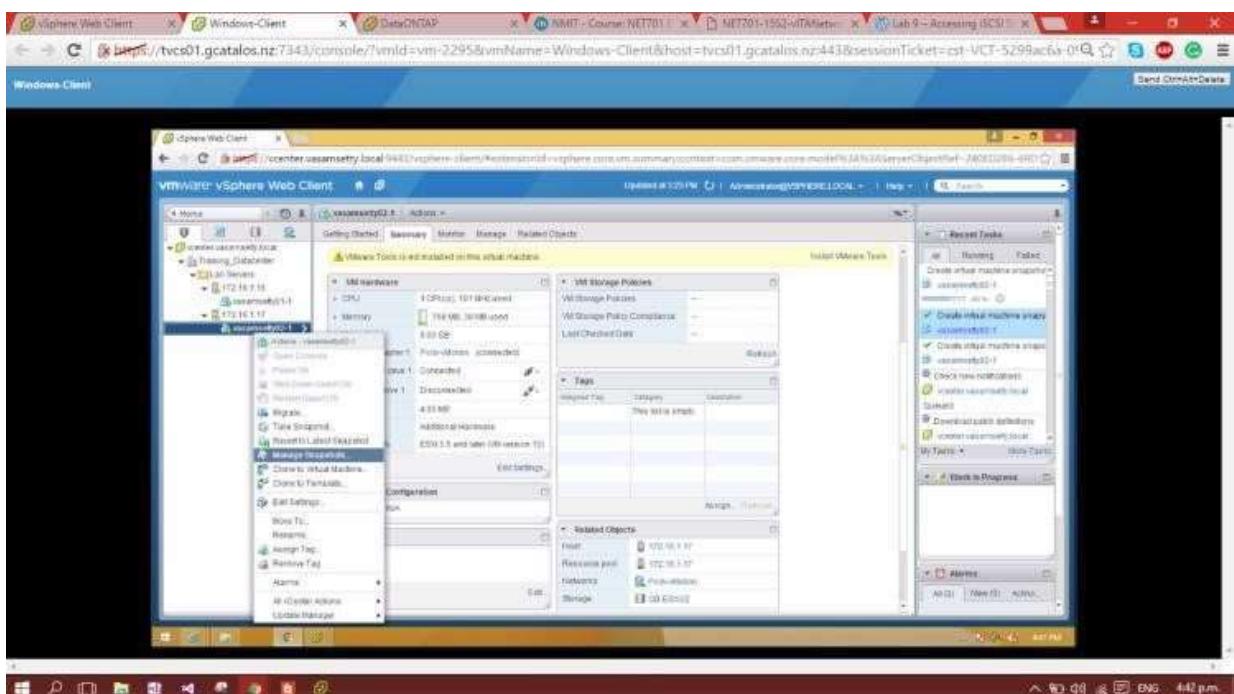
I then deleted the cpubusy.vbs file and created another snapshot of the virtual machine with the following settings:



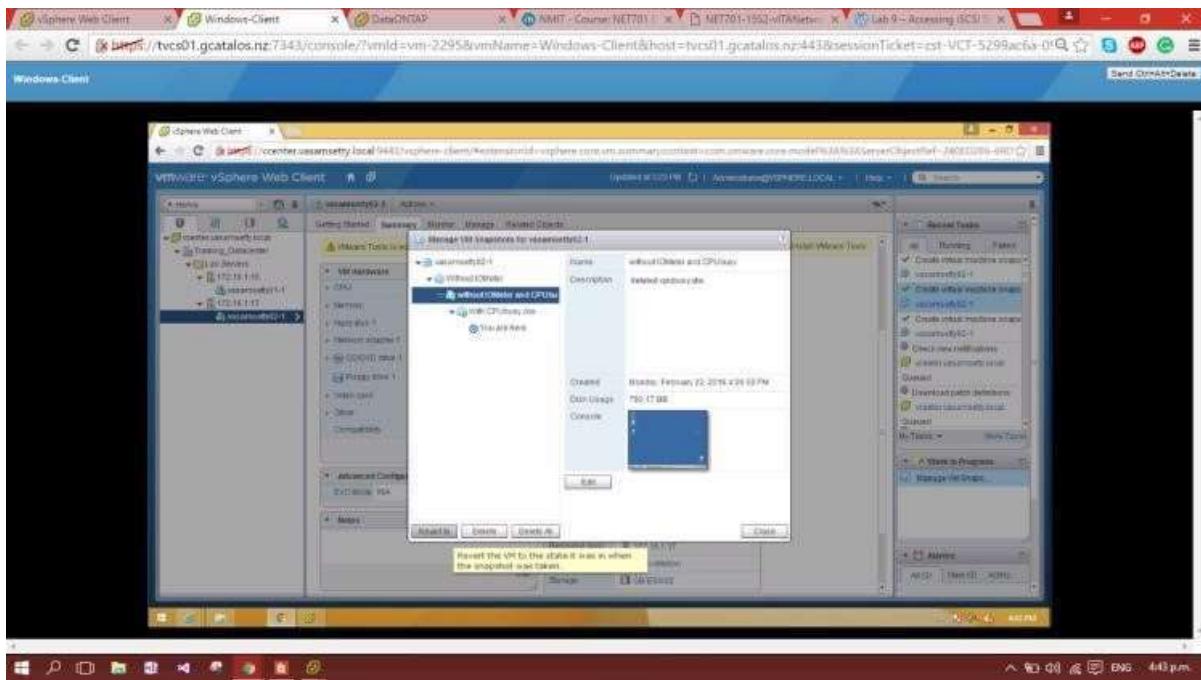
I then copied the cpubusy.vbs file back from my ‘Downloads’ folder to the desktop and created another snapshot with the following settings. Note that I have left the ‘Snapshot the virtual machine’s memory’ tick box ticked this time.



After the third snapshot had been created, I right-clicked the ‘vasamsetty01-2’ virtual machine and selected ‘Manage Snapshots…’.

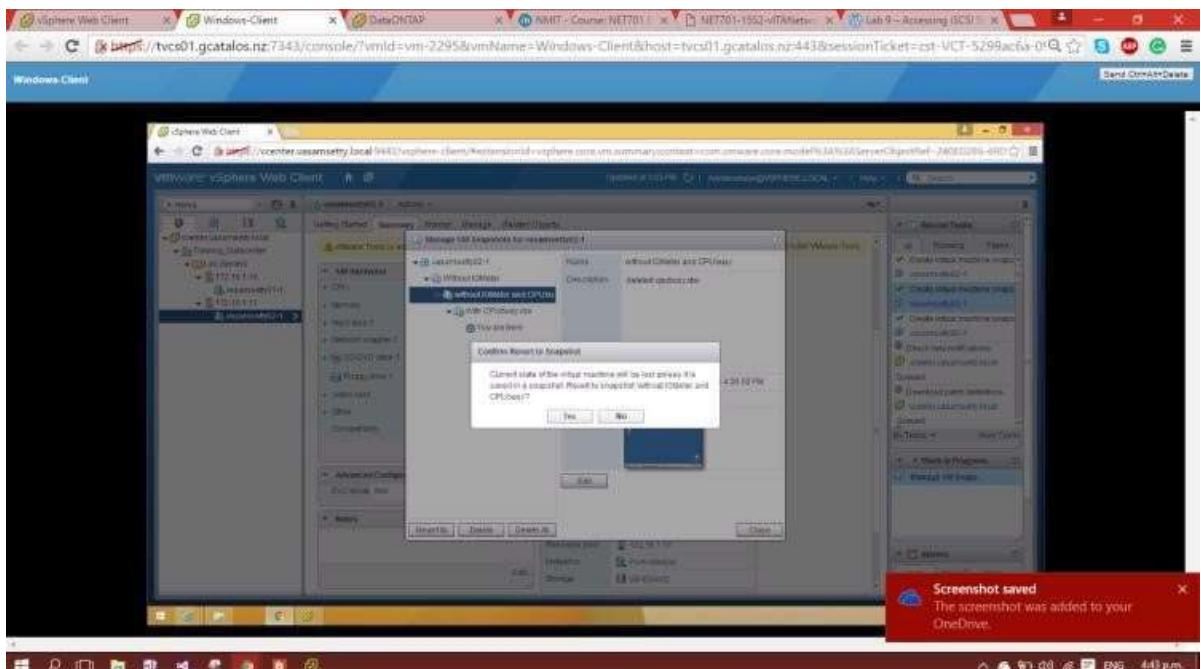


This opened up the snapshot manager where I could see the three snapshots that I had created.

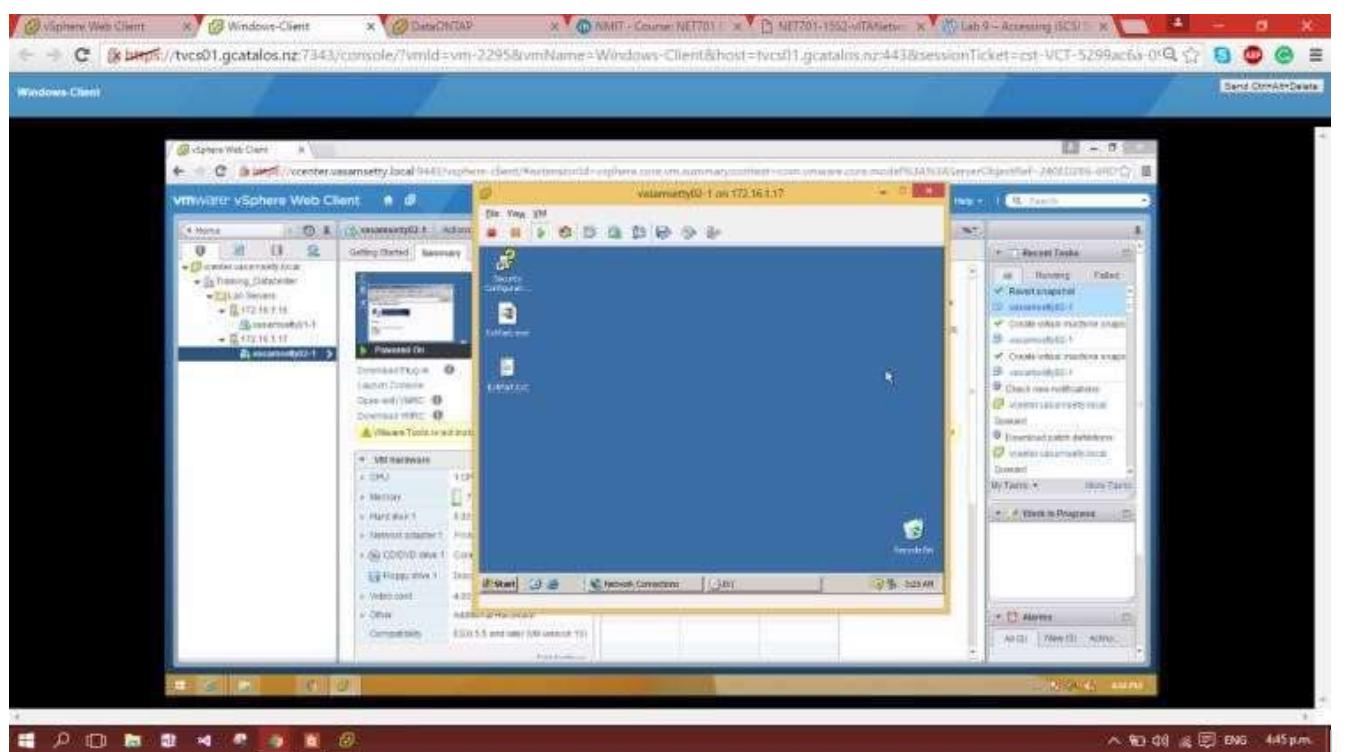
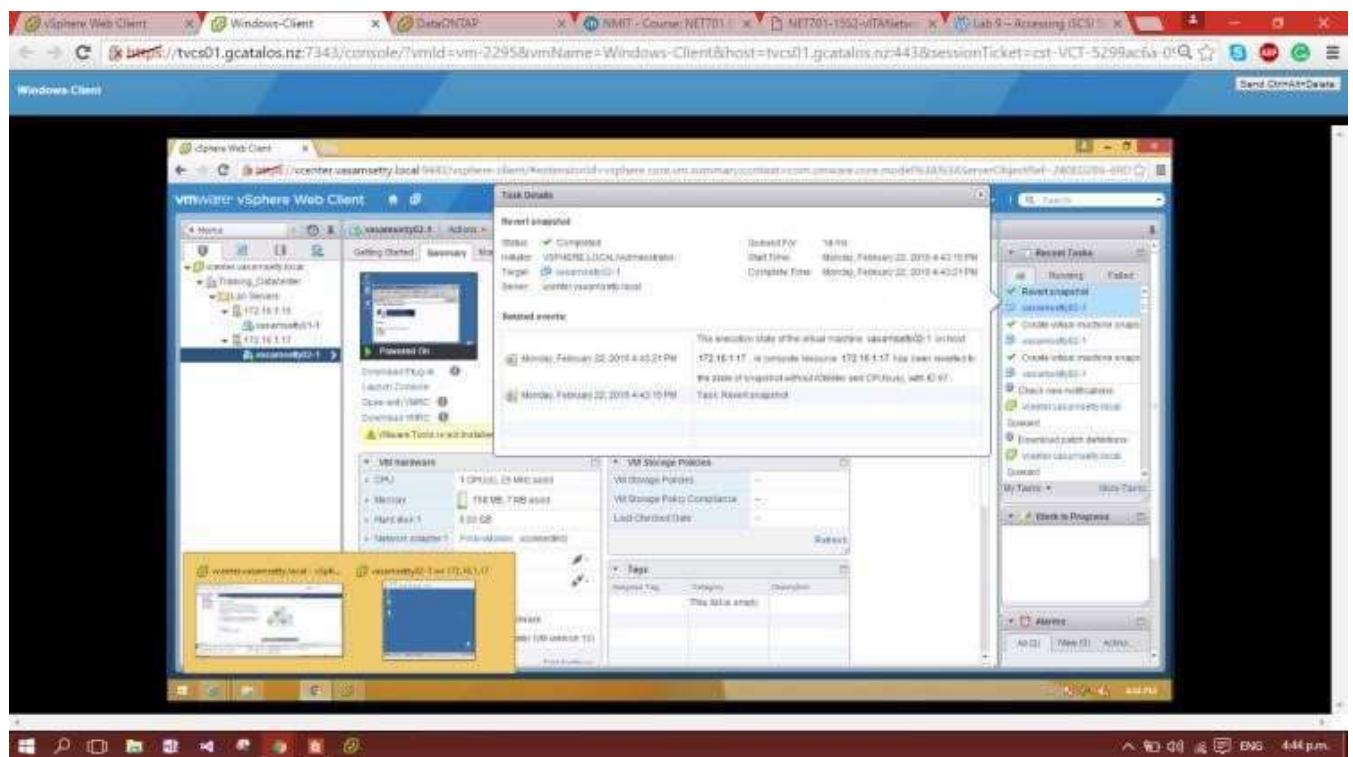


### Reverting to a Snapshot

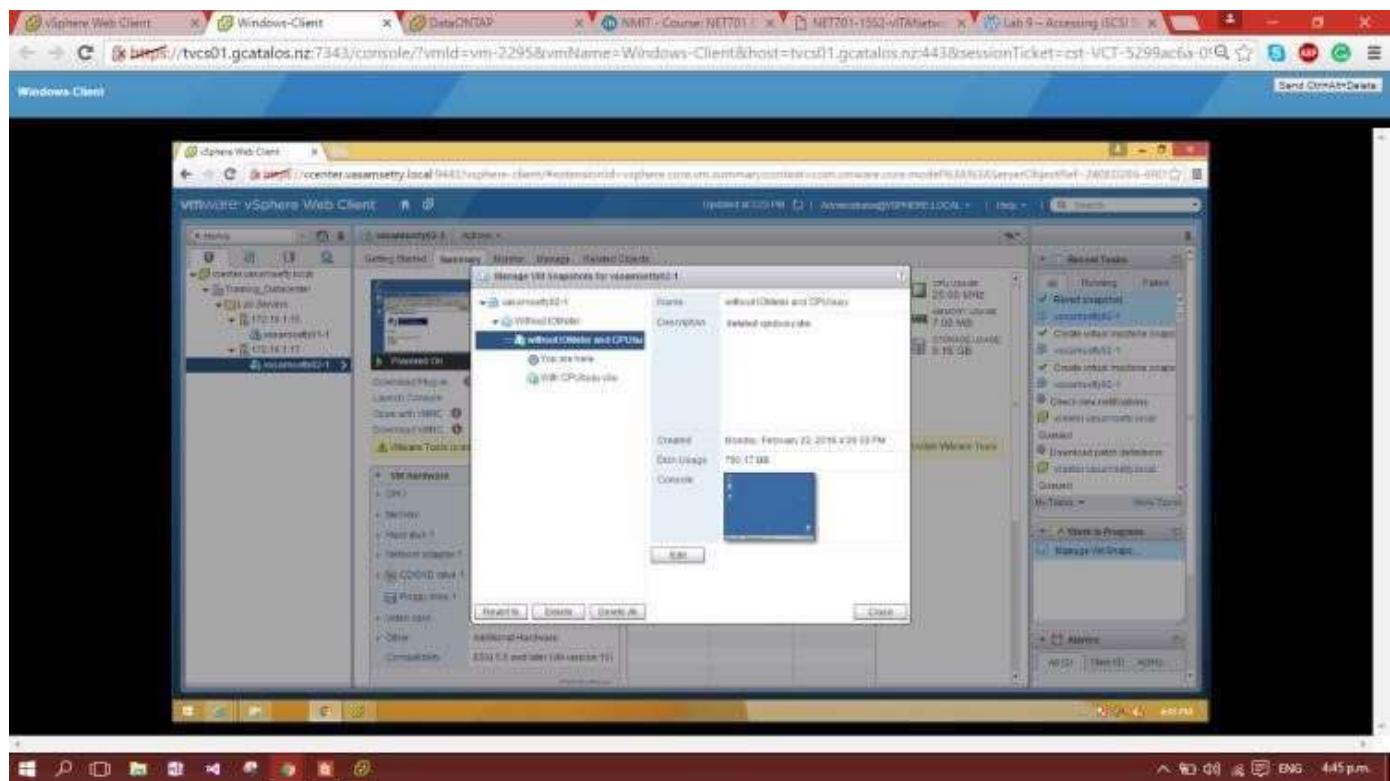
For this task i re-opened Snapchat Manager by selecting on my vasamsetty02-1 VM and selected without IOmeter and CPUbusty.vib. and clicked the 'Revert' button in order to revert to that snapshot.



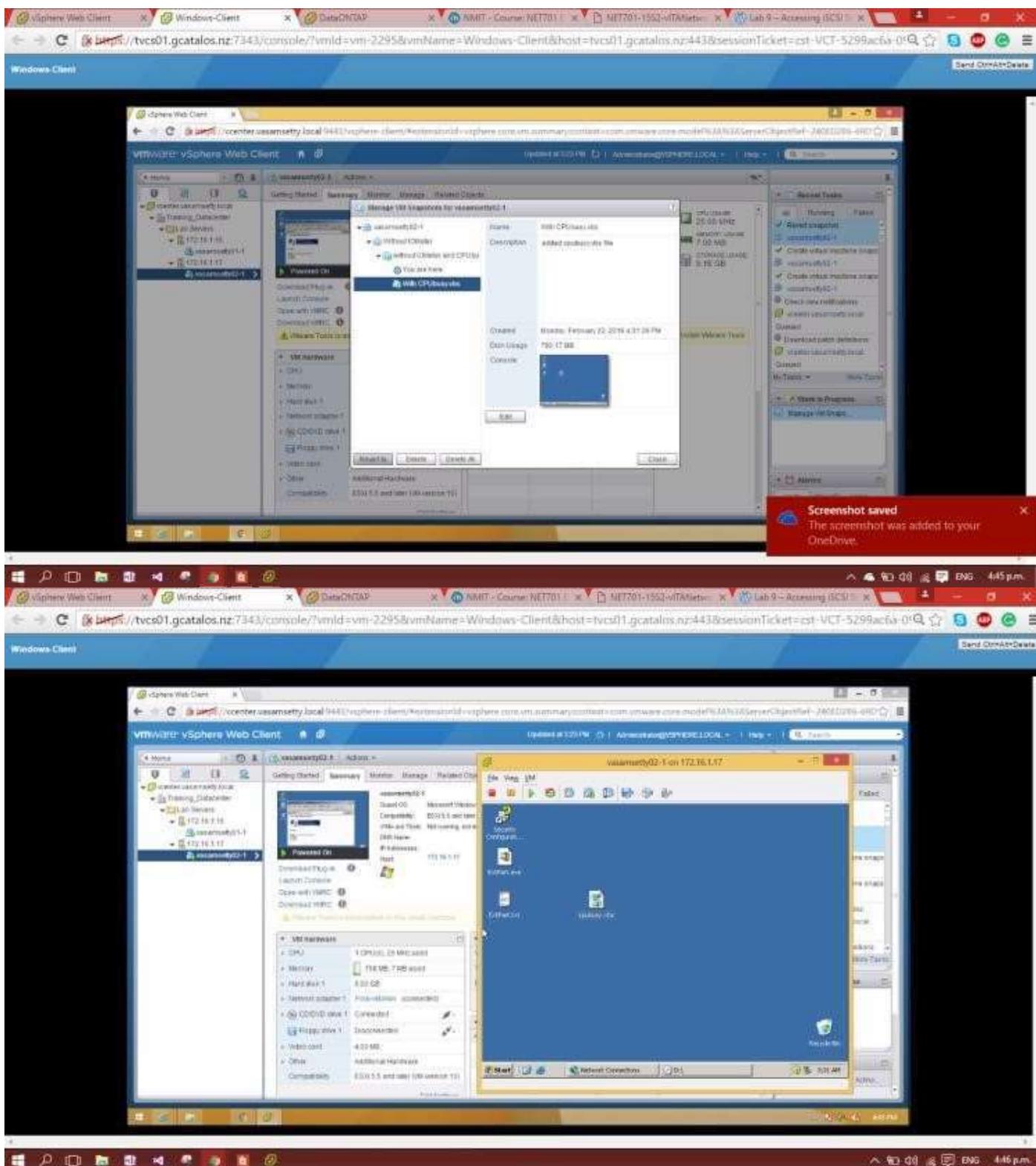
I see the task accomplished in the recent tasks menu.This states that my VM is reverted back to the time in which we deleted the IOmeter and cpubusy.vib files



After that, I returned to the snapshot manager for the 'vasansetty02-1' virtual machine and noticed that the 'You are here' indicator had shifted to below the 'Without iometer and cpubusy' snapshot.

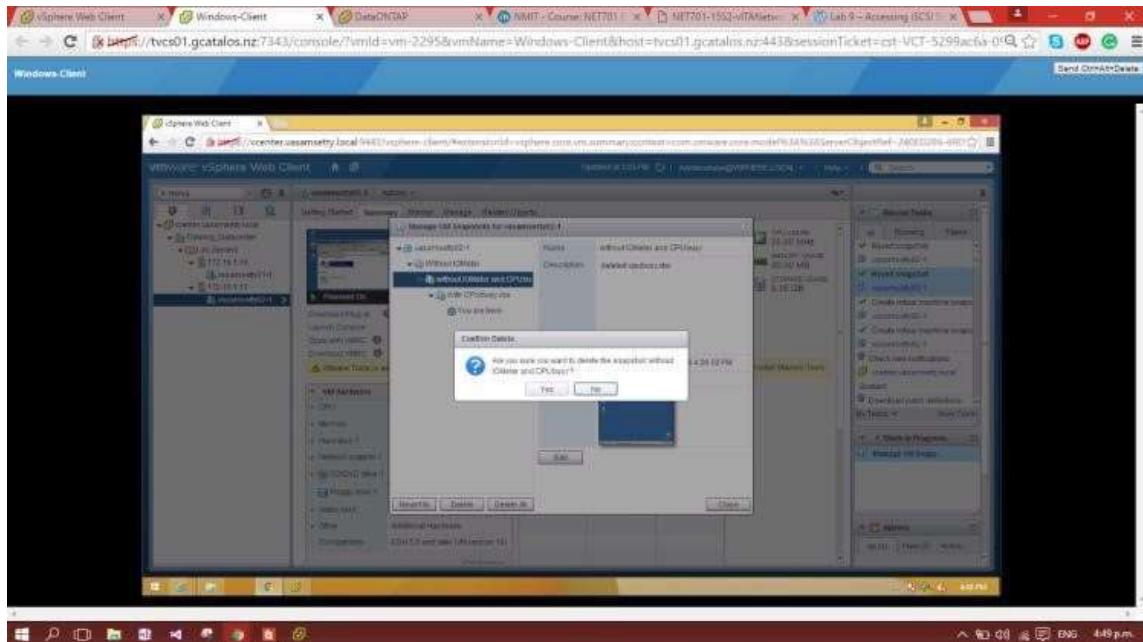


I then reverted to the 'With cpubusy' snapshot. I noticed that the virtual machine did not get powered off this time. This was probably because I ticked the 'Snapshot the virtual machine's memory' tick box when I created this snapshot and so it preserved that state that the virtual machine was in at the time of the snapshot. Finally, I checked to see that only cpubusy.exe had returned to the desktop.

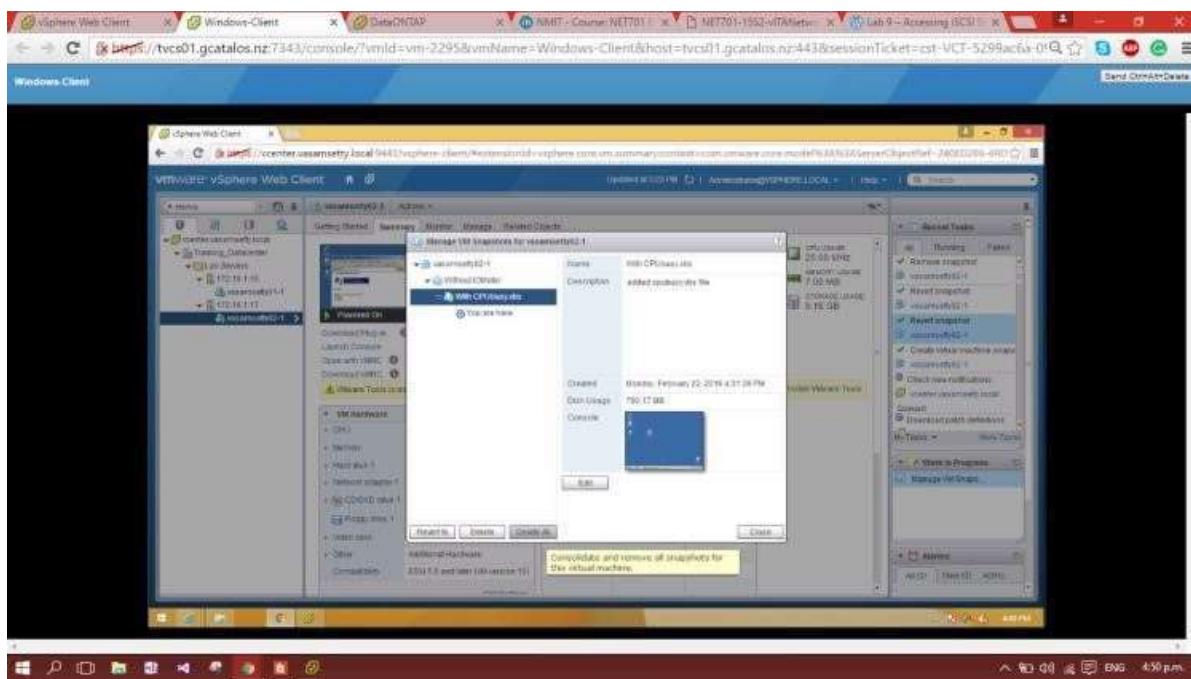


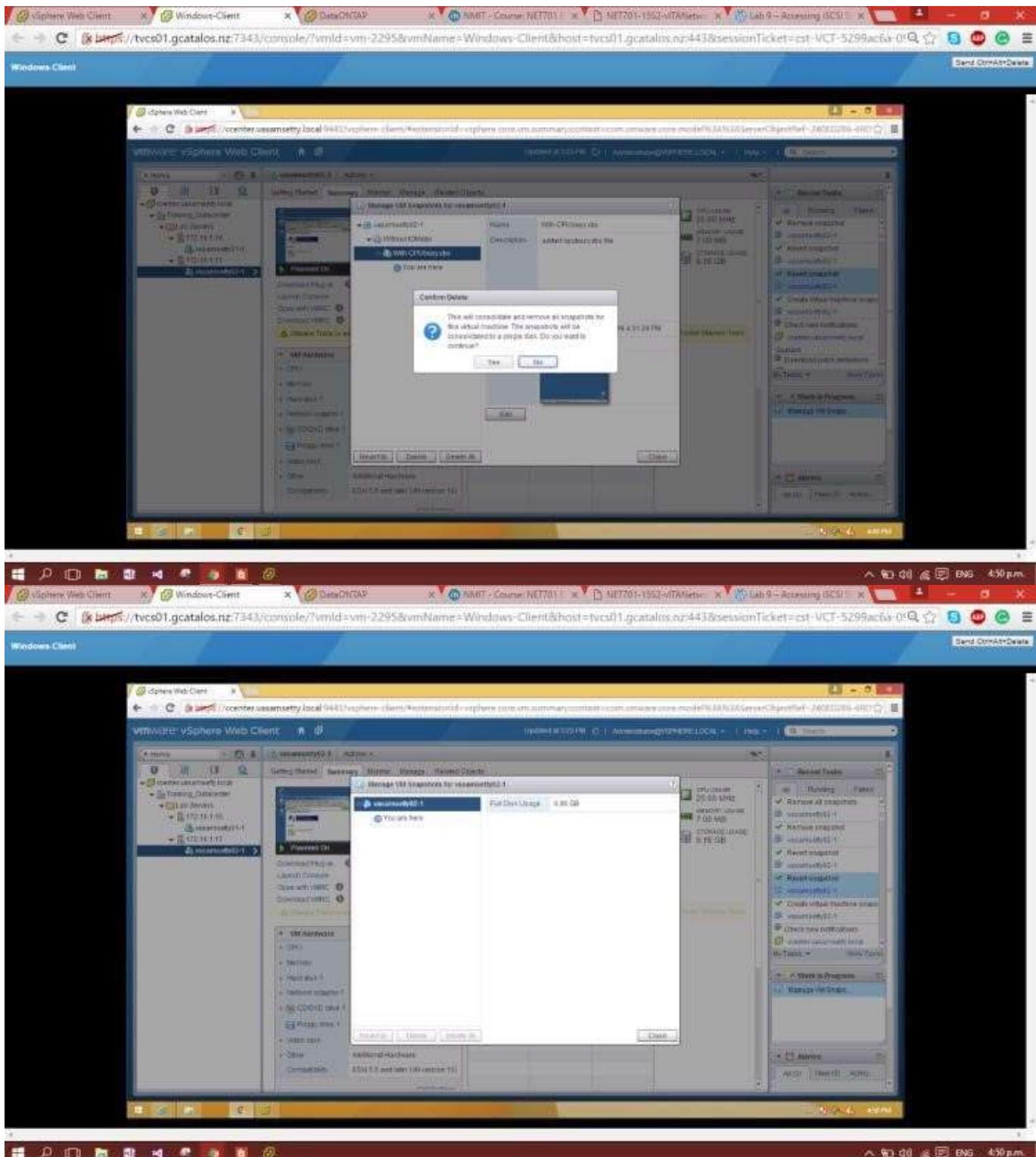
### Deleting an Individual Snapshot

For this task i once again navigated to Snapshot Manager on my vasamsetty02-1 VM noticed my you are here position selected that and clicked on DELETE we get an warning pop-up and clicked on yes.

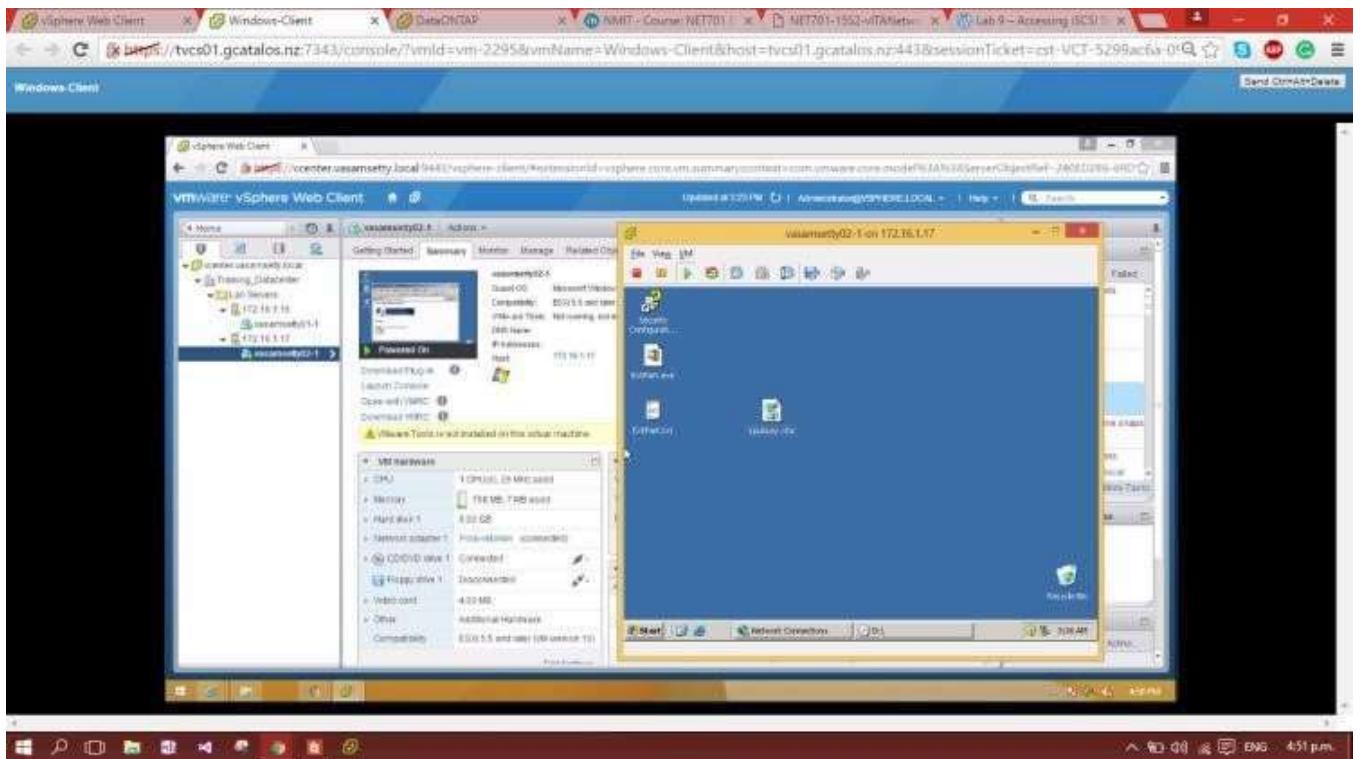


I can see my selected snapshot is deleted successfully. I did the same task once again to delete with CPUBUSY.vib snapshot and am successful.





.Finally, I checked to see whether cpubusy was still on the desktop of the virtual machine and found that it was.

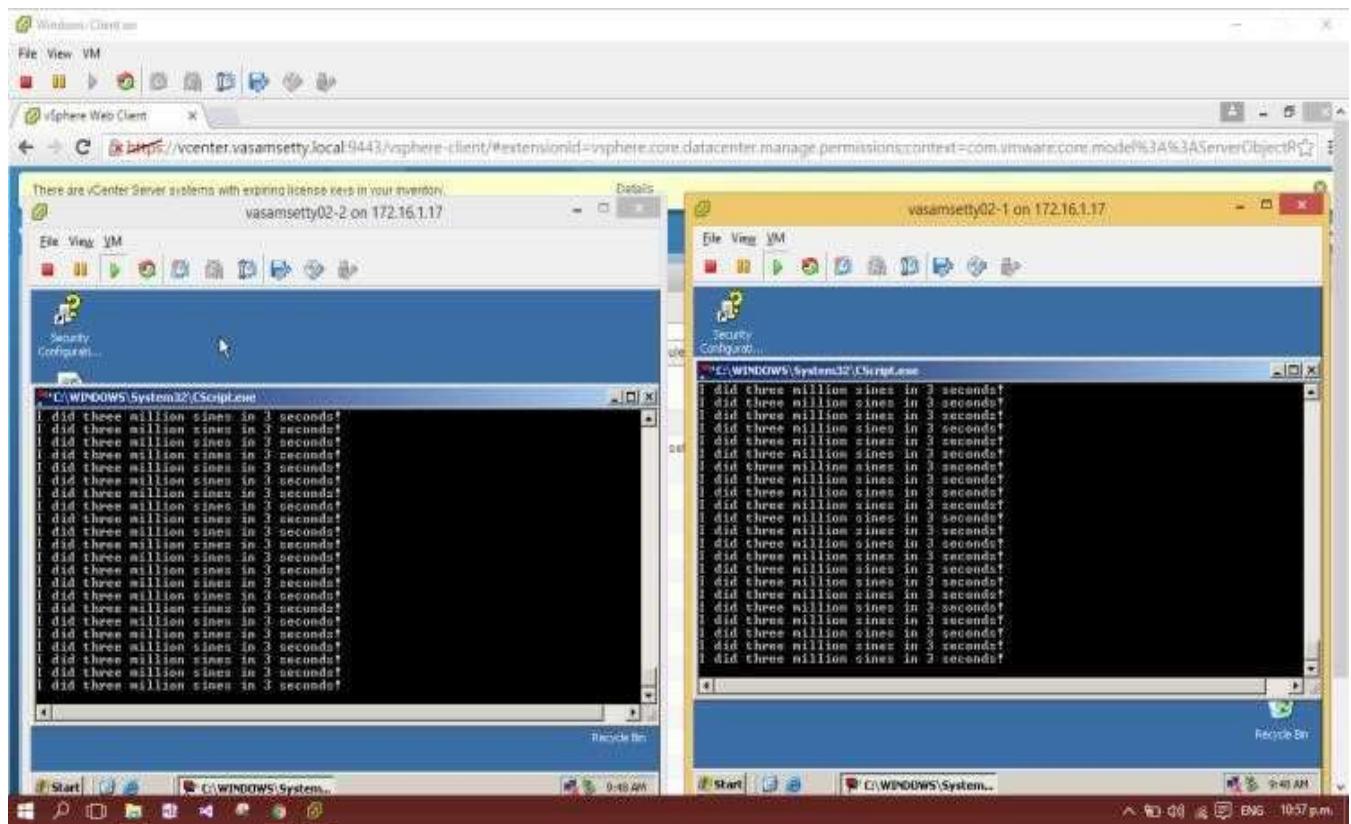


Overall playing with the snapshots i found that deleting the snapshots or managing the snapshots will move the virtual machine from one state to its previous state. But it won't applies to manage the current state(you are here) position.

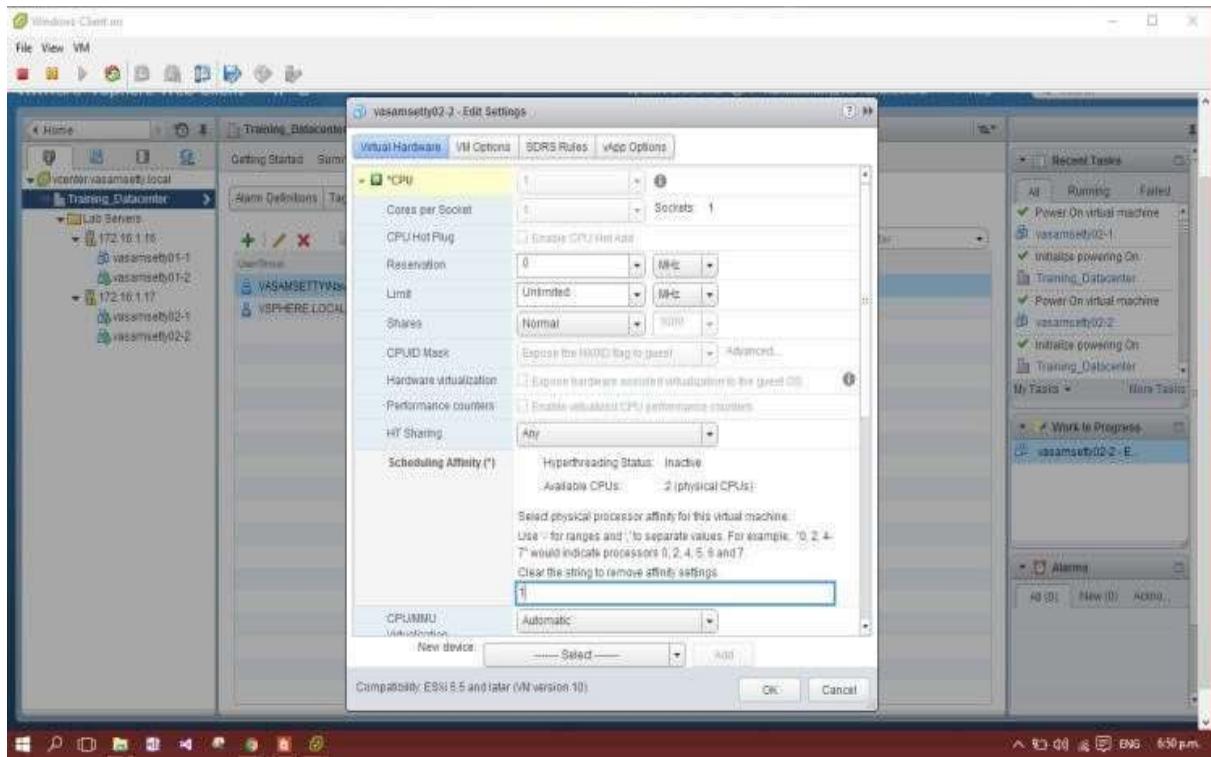
## Practical No 12 :- Managing Resourcing Pools

### Creating CPU Contention

To begin this task, I logged into the vSphere Web Client, navigated to 'Home -> vCenter -> VMs and Templates', powered on both of my virtual machines, logged in to both of them, and opened cpubusy.vbs via the command prompt in order to create some CPU activity. It appeared to stabilise at around three seconds.



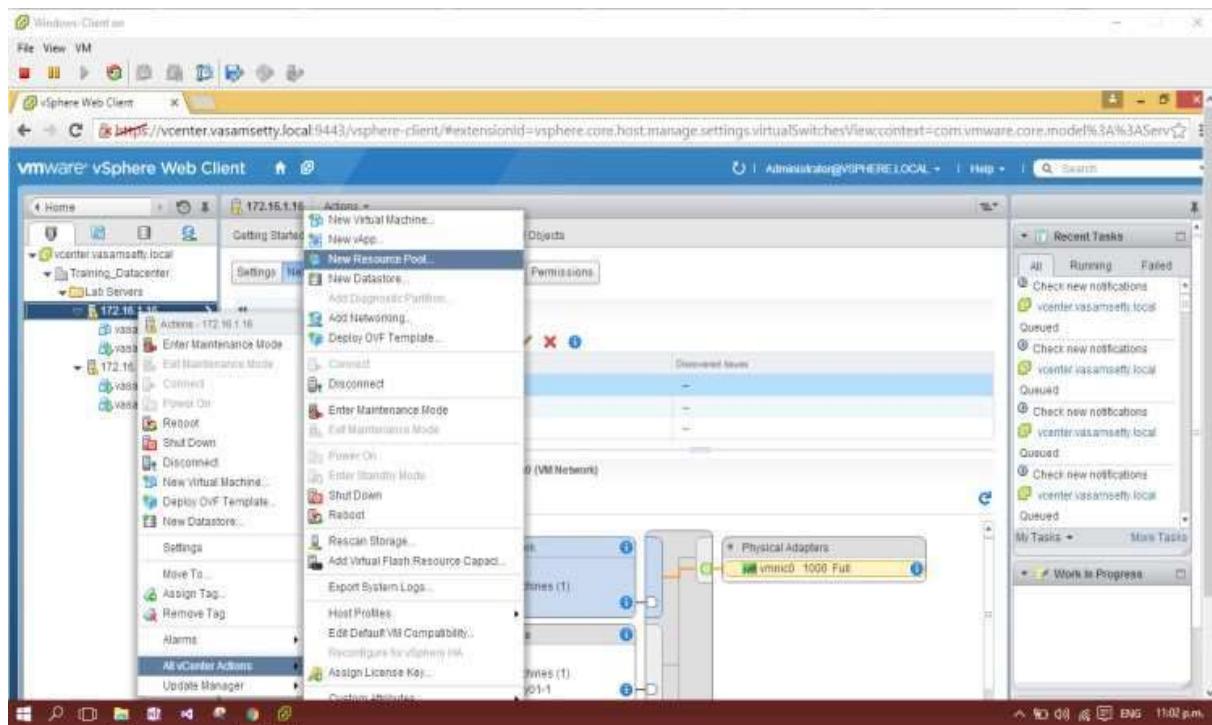
As per my task i edited the settings of both the virtual machines and changed the "Schedule Affinity" to "1"



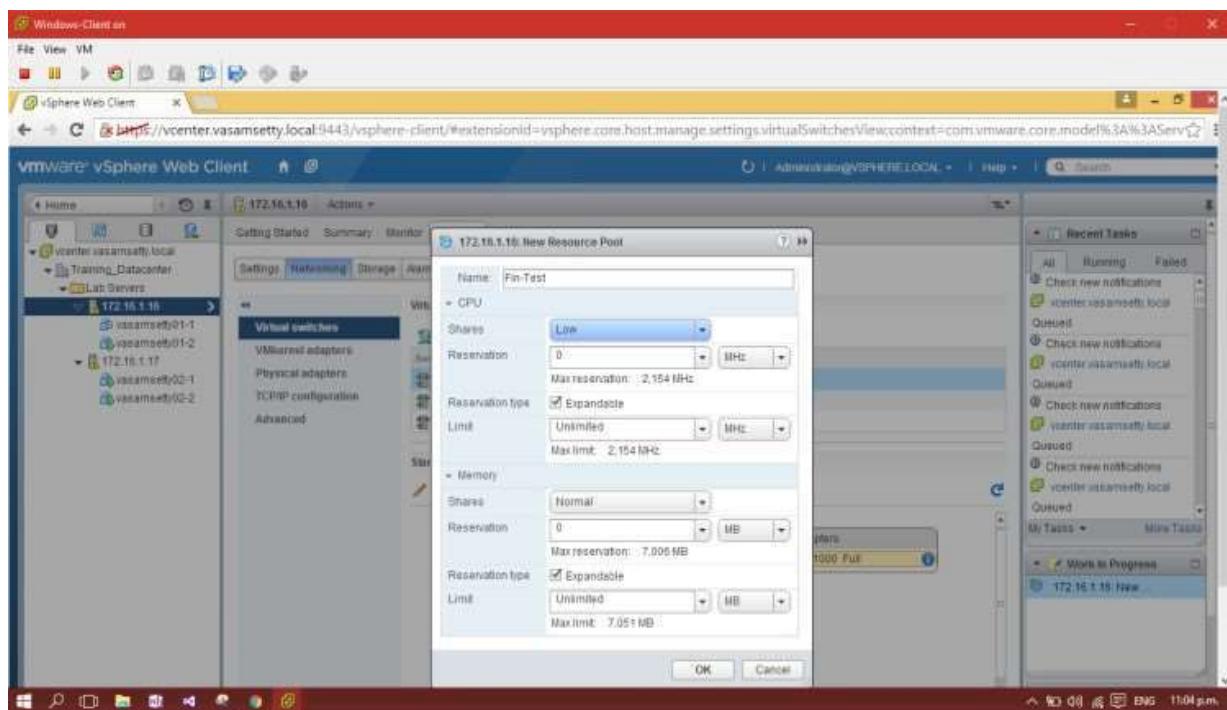
and again i navigated to my virtual machines consoles and found the no difference in showing the time. They just appeared a little difference between the two VM's. Overall the average time taken by both the VM's is exactly 3 seconds.

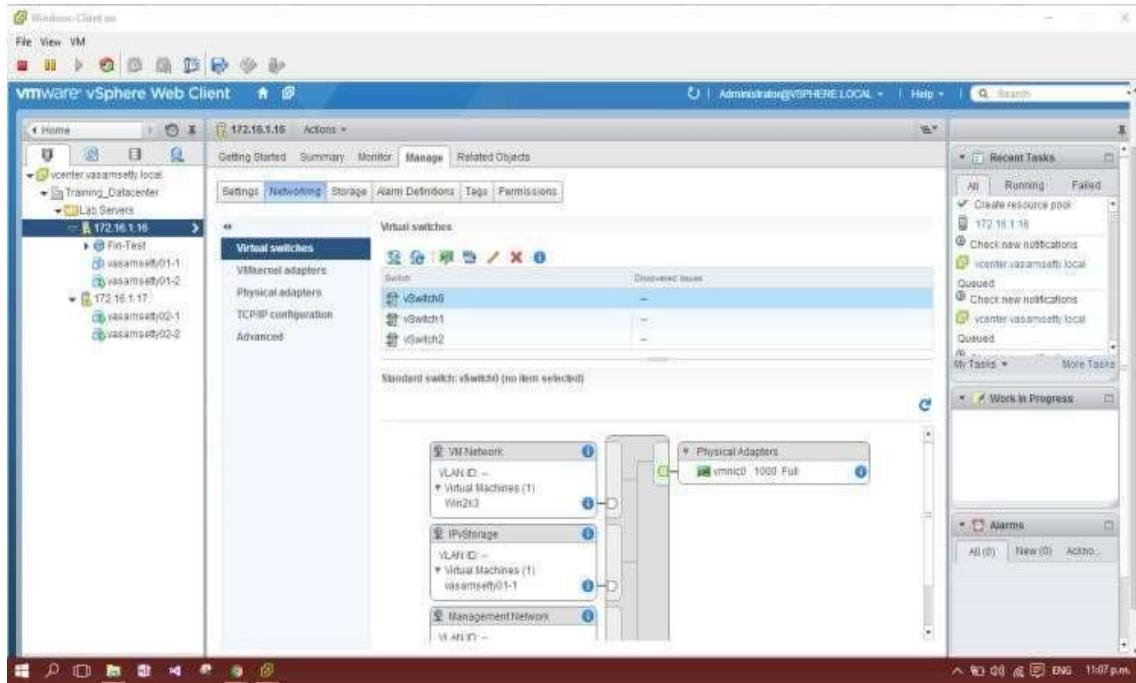
### Creating a Resource Pool's Named Fin-Test and Fin-Pod

Creating a resource pool is an easy task, I just navigated to the Vcenter->Hosts and Clusters', right-clicking my first ESXi host, and selecting 'All vCenter Actions -> New Resource Pool'. The 'New Resource Pool' wizard then opened.



For the name, I entered 'Fin-Test'. For the CPU Shares level, I selected 'Low'. I left the remaining settings alone and created the resource pool.

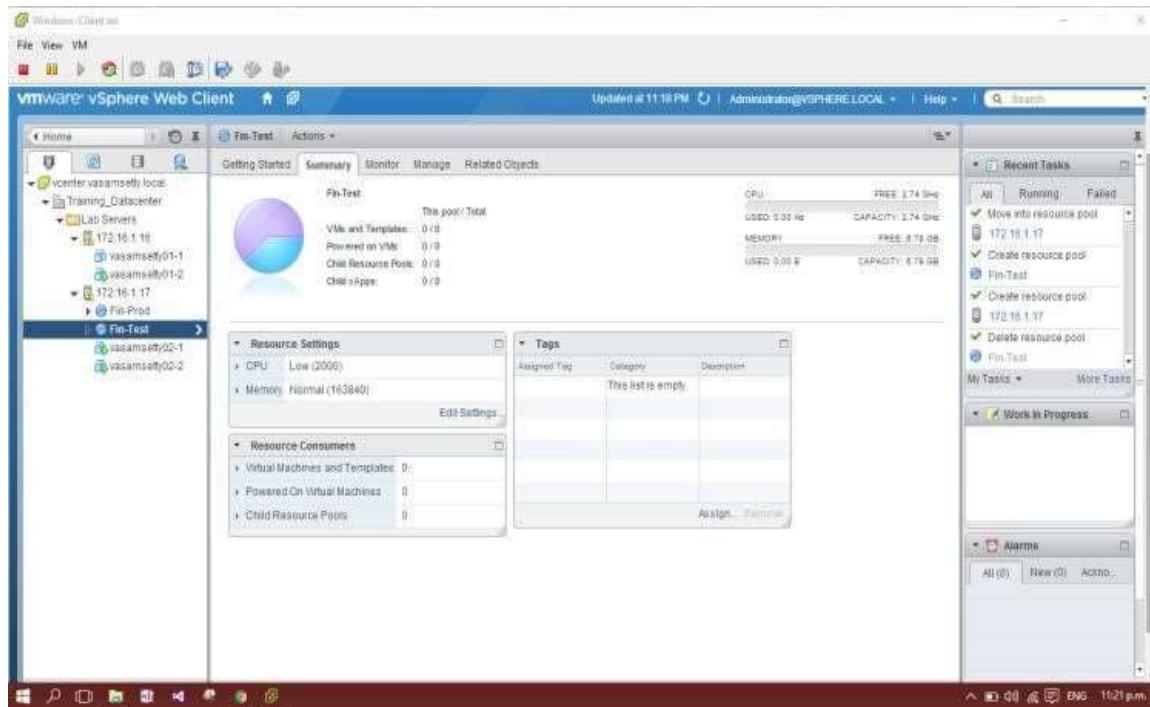




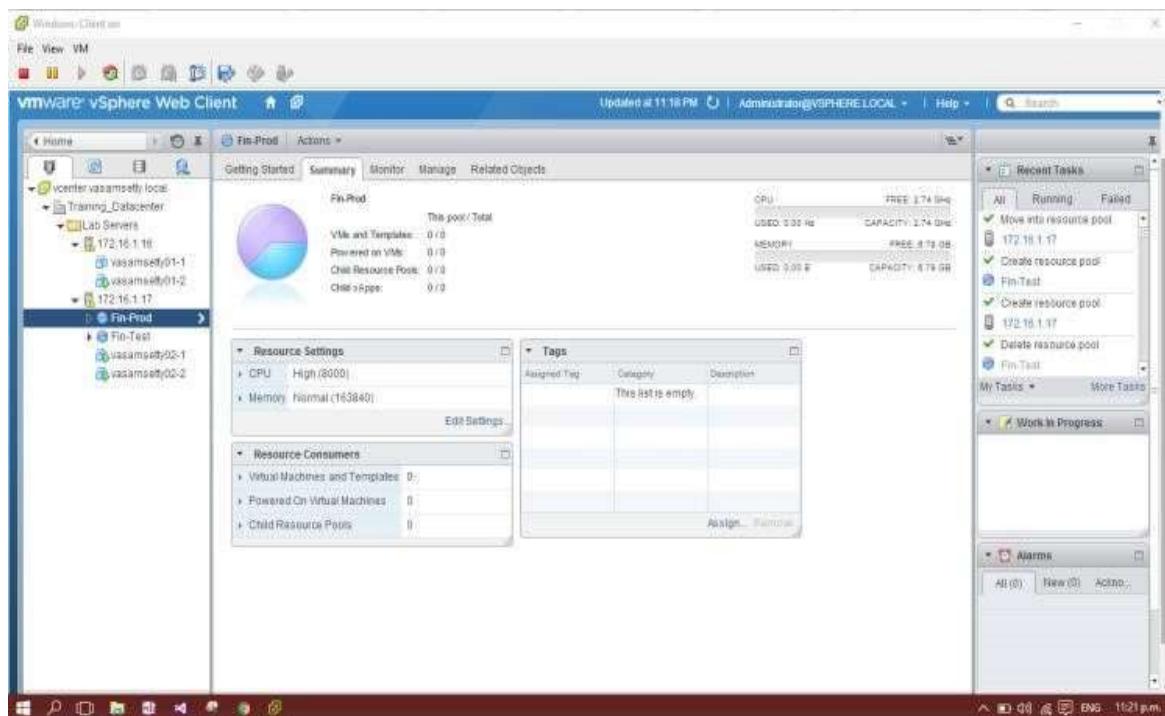
Similarly i added one more resource pool For the name, I entered ‘Fin-Pod’. For the CPU Shares level, I selected ‘High’. I left the remaining settings alone and created the resource pool.

### Verifying Resource Pool Functionality

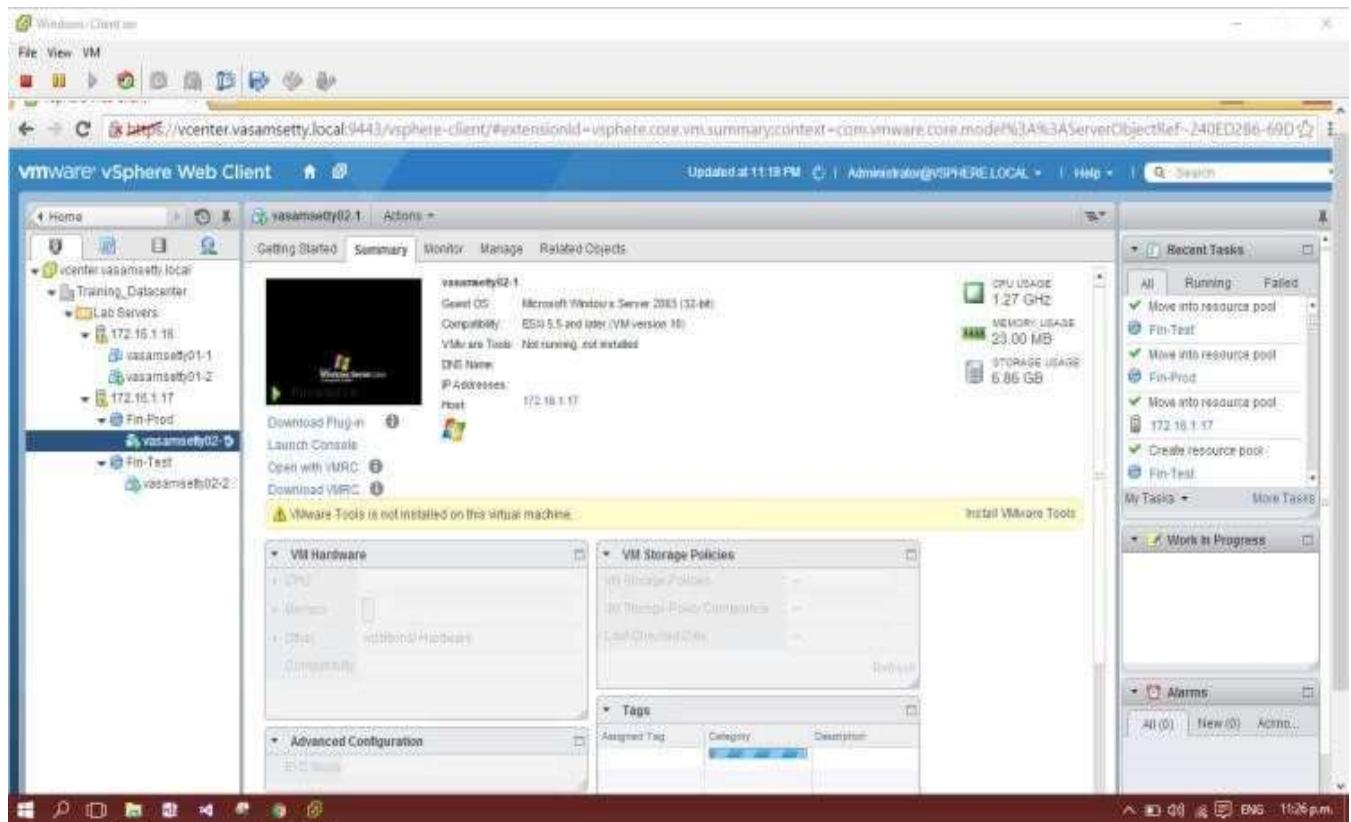
To begin this task, I selected the ‘Fin-Test’ resource pool from the inventory and clicked the ‘Summary’ tab. I noted that the available CPU shares for this resource pool were 2,000.



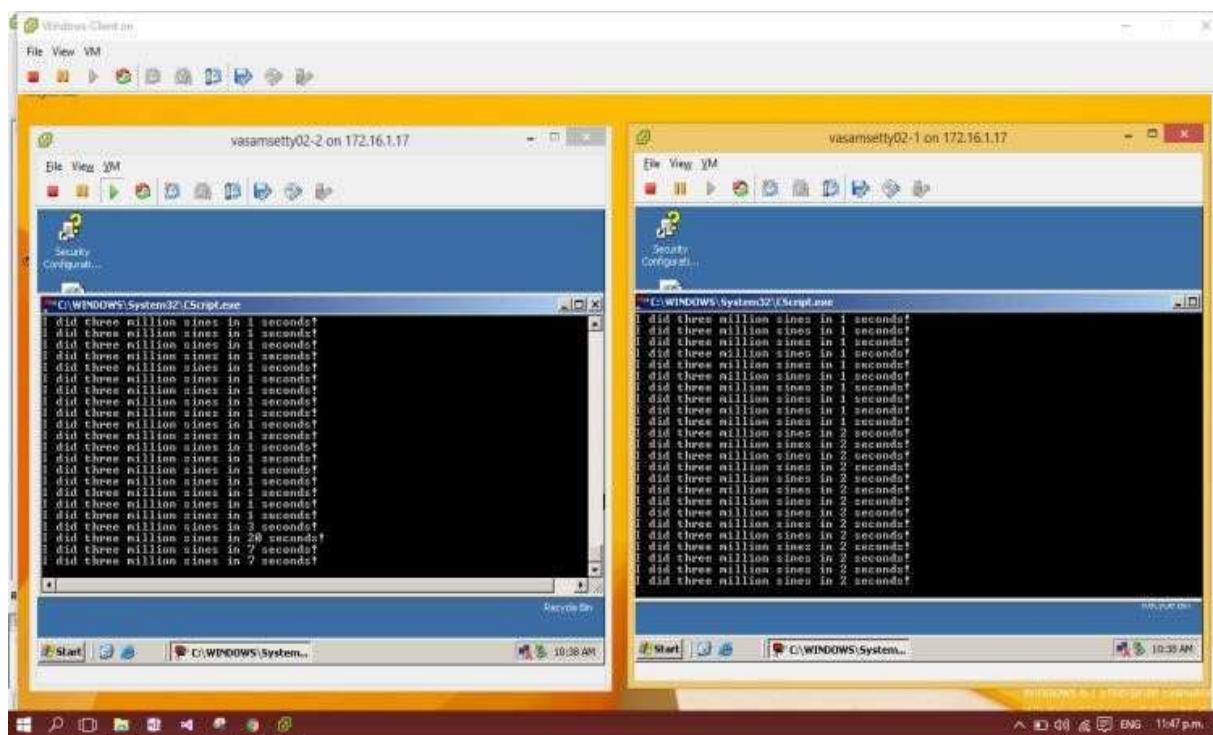
I then selected the 'Fin-Prod' resource pool and noted it had an available CPU shares amount of 8,000.



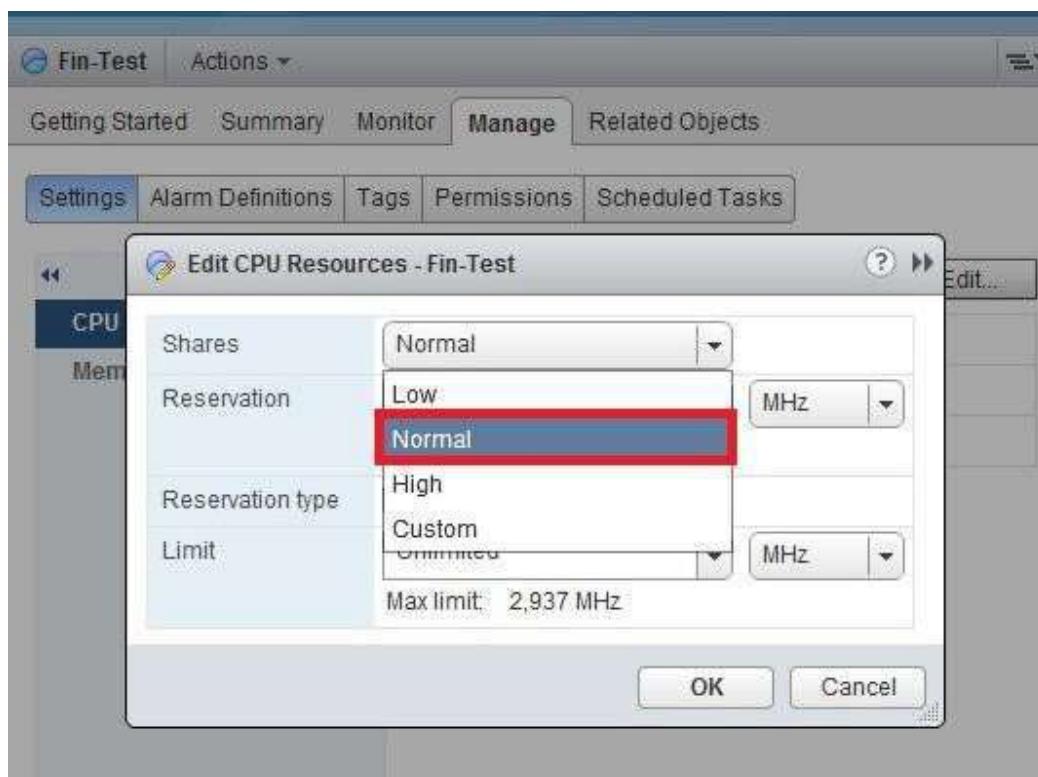
to check the resource pool functionality i moved both VM's to each resource pool opened the consoles and run the cpubusy.vib file and checked the average time taken by both of the vm's



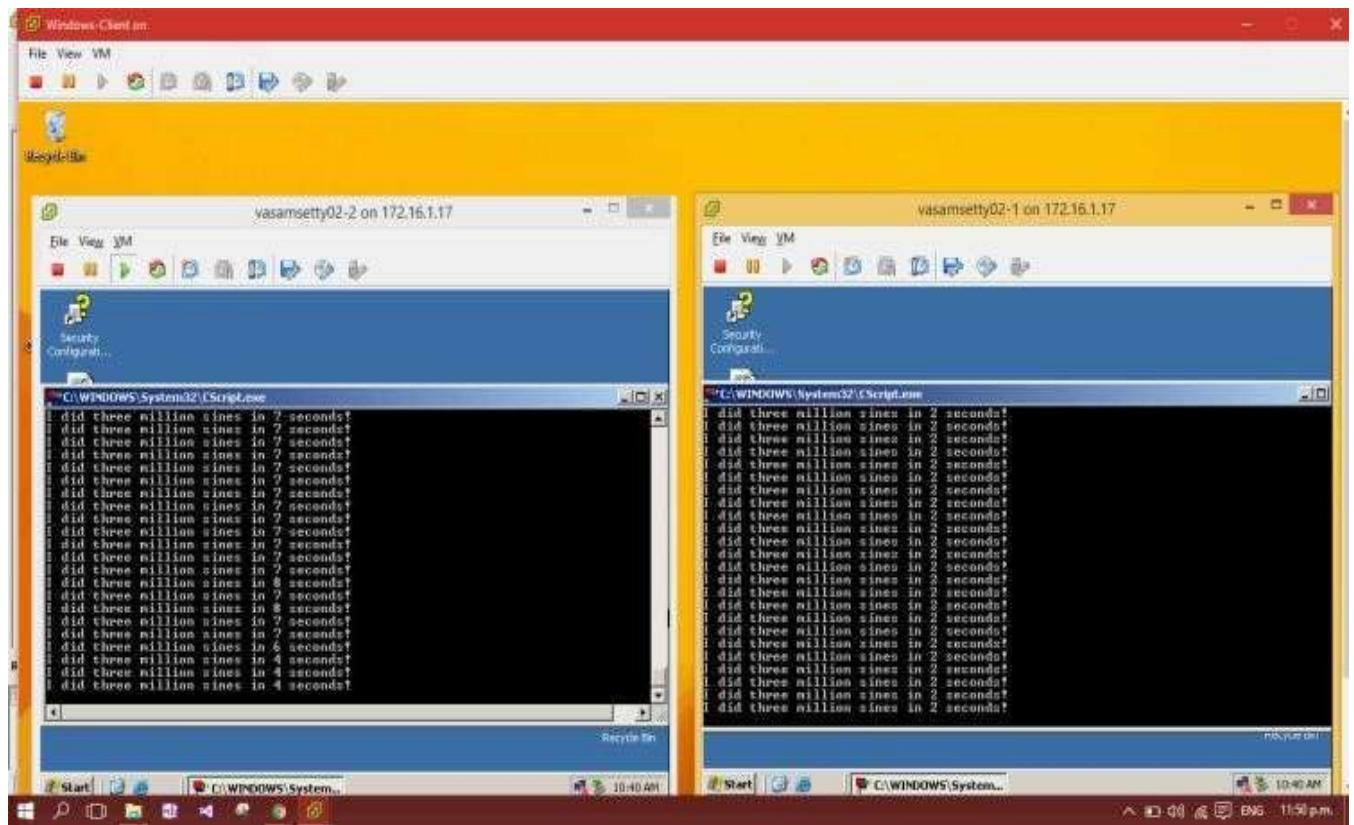
The 'vasamsetty02-1' virtual machine in the resource pool with 8,000 shares consistently achieved around two seconds. where as the 'vasamsetty02-2' virtual machine in the resource pool with only 2,000 shares makes many changes like it is altering its time and finally sets around seven seconds.



I then changed the CPU shares of the 'Fin-Test' resource pool from low to normal...



and compared the results of the script again. The ‘vasamsetty02-1’ virtual machine remained at two seconds while the ‘vasamsetty02-2’ virtual machine dropped from seven seconds to four seconds.



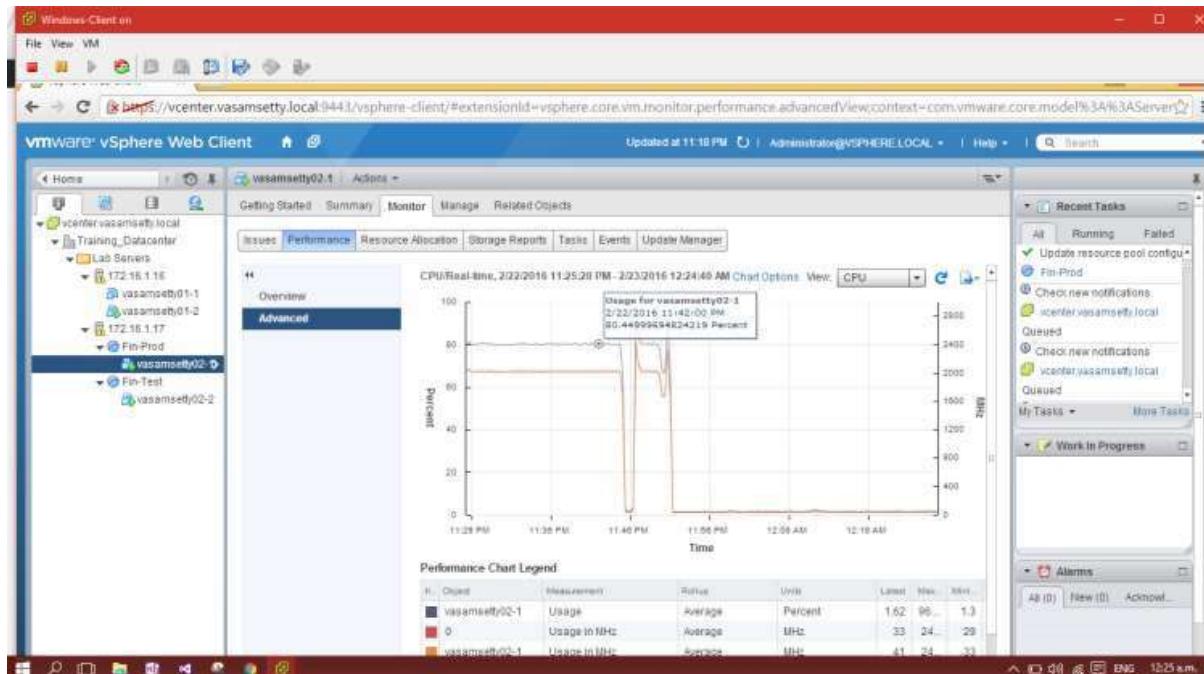
## Practical 13: Monitoring Virtual Machine Performance

In this lab, I was required to do the following tasks:

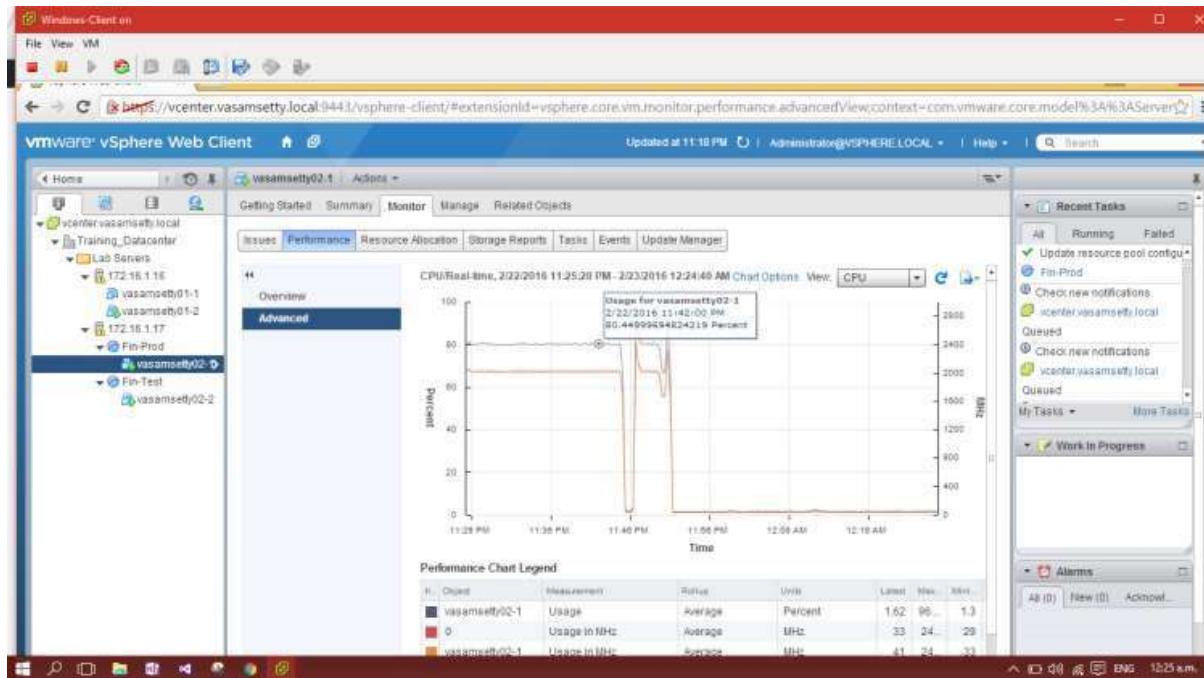
- Create CPU Activity
- Use vSphere Web Client to Monitor CPU Utilization
- Undo Changes Made to the Virtual Machines

### Creating CPU Activity

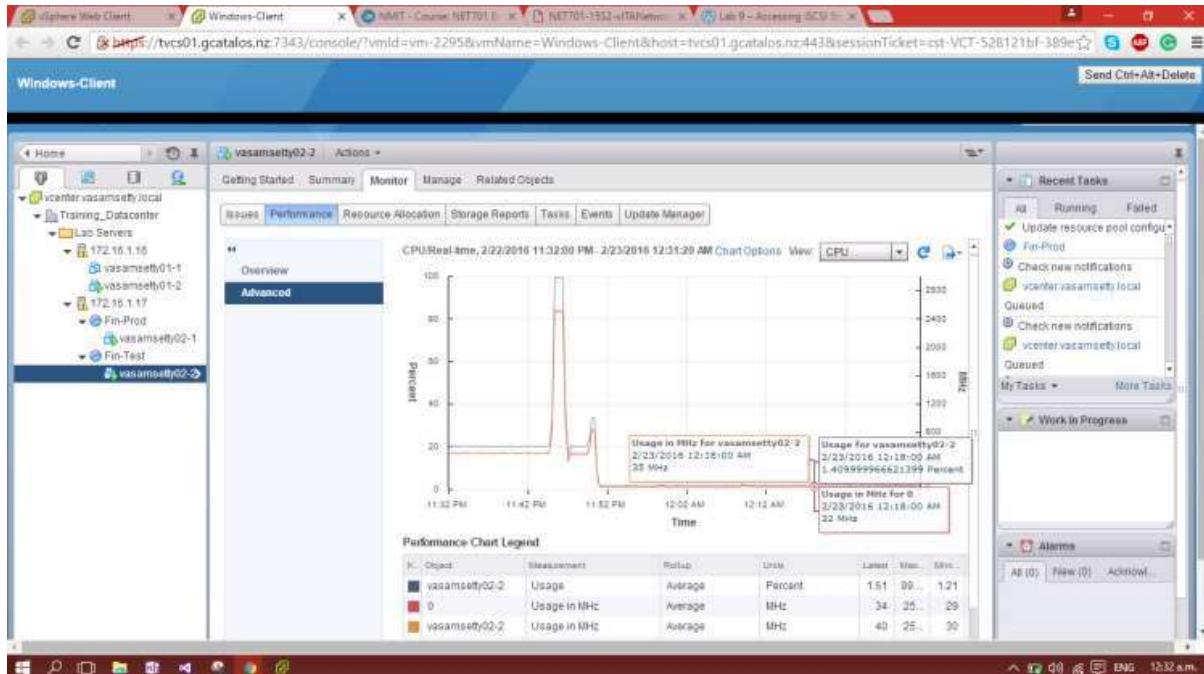
- I had done this task many times in my previous lab. I created a CPU activity by opening my vm consoles and started up the cpubusy.vb script on both of them.
- **Using vSphere Web Client to Monitor CPU Utilization**
- in this task i navigated to one of my VM and navigated to the ‘Monitor -> Performance -> Advanced’ tab. This displayed a graph of CPU usage.



I then opened a second instance of the vSphere Web Client and configured the same chart options for my ‘vasamsetty02-2’ virtual machine. After that, I obtained the CPU ready value for both virtual machines (with the scripts still running on both machines). vasamsetty02-1 CPU ready value:



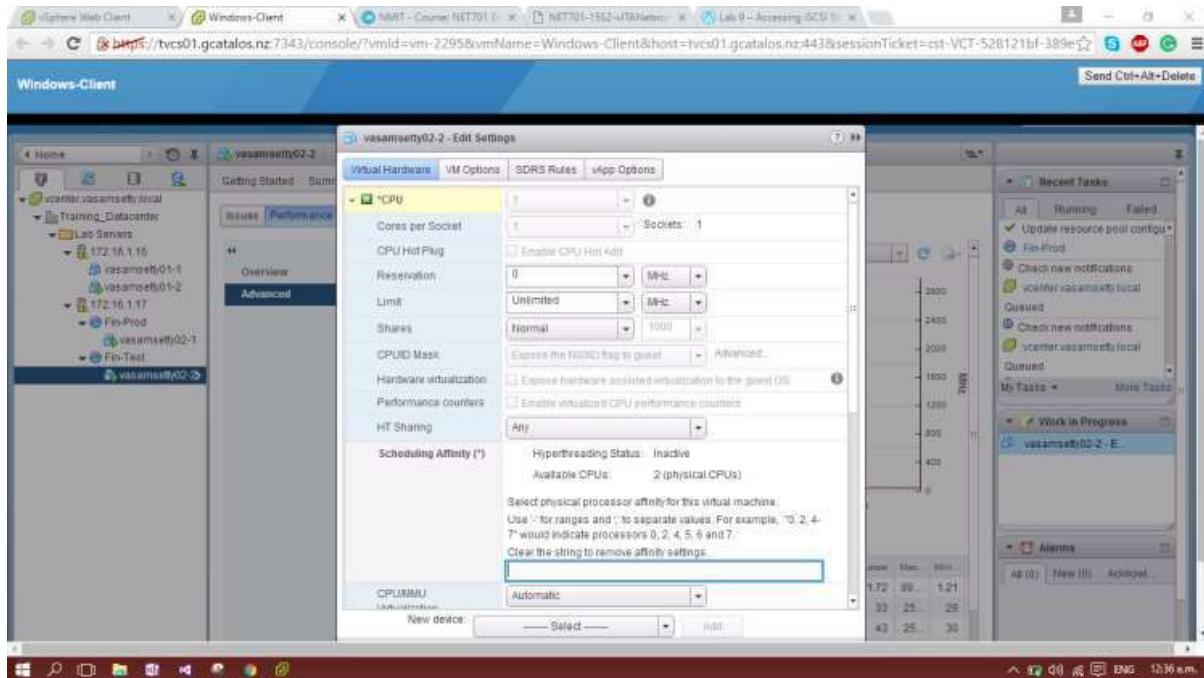
I then stopped the cpubusy.vbs script on both of the virtual machines and compared the values again. Both machines took a significant dip in their milliseconds value. I imagine that this is because there was no longer a significant strain on the CPU as the script had been disabled. Jamie01-2 CPU ready value:



## Undoing Changes Made to the Virtual Machines

This task simply required that I deleted the CPU affinity value that I had set earlier for both of

my virtual machines. In order to do this, I right-clicked each virtual machine, selected ‘Edit settings…’, expanded the CPU tab, and removed the ‘1’ value.



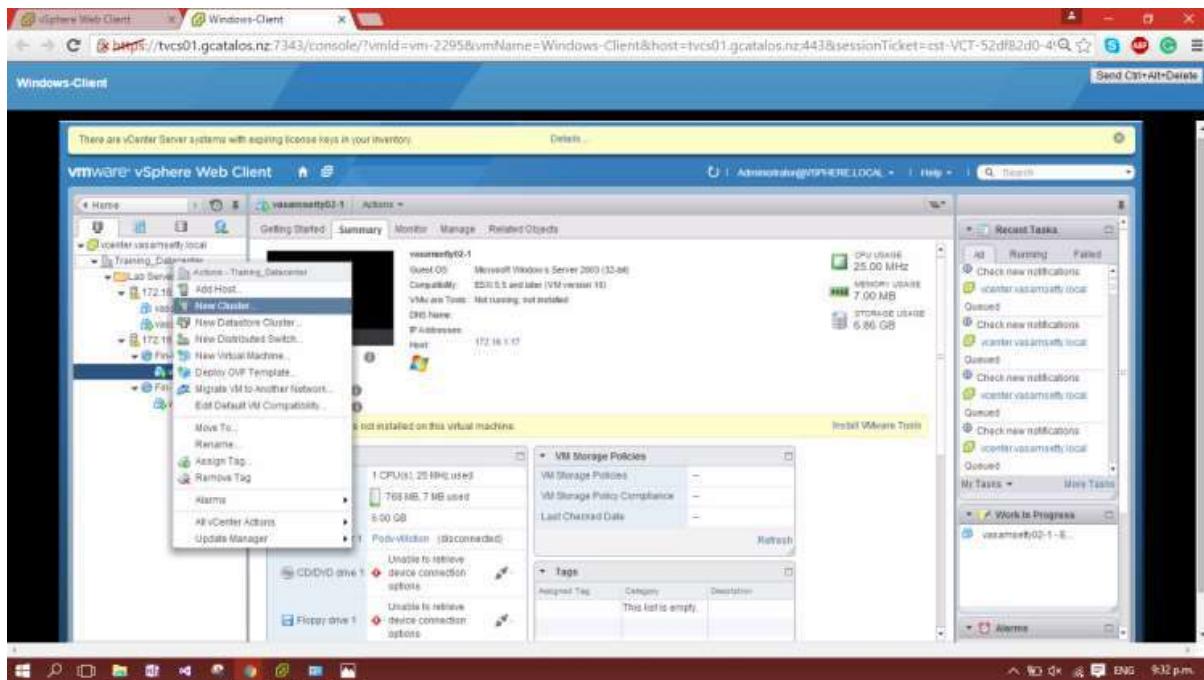
## Lab 14 : Using vSphere HA..

In this lab am gonna do the lab task's.

- Create a Cluster Enabled for vSphere HA
- Add Your ESXi Host to a Cluster
- Test vSphere HA Functionality
- Determine the vSphere HA Cluster Resource Usage
- Manage vSphere HA Slot Size
- Configure a vSphere HA Cluster with Strict Admission Control
- Prepare for Upcoming Labs

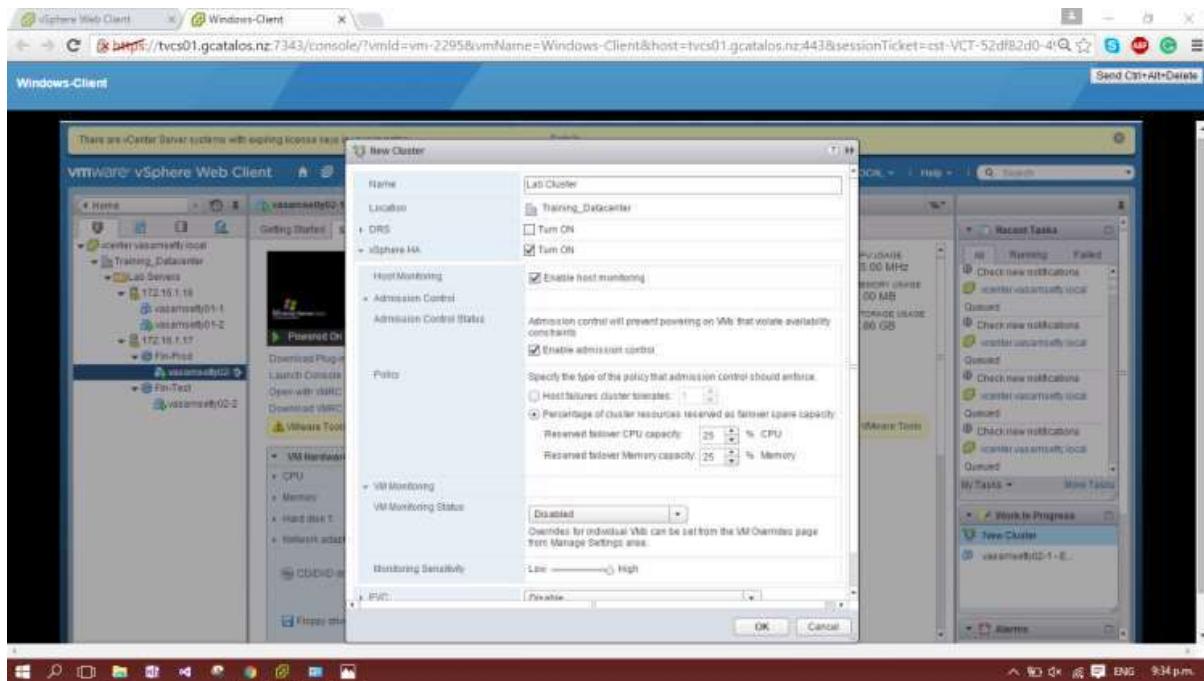
### Creating a Cluster-Enabled for vSphere HA

For creating a cluster i logged into my Vsphere Web Client, and navigated to my ‘Training-Datacenter’ datacenter and right clicked on it and selected on ‘New Cluster’ from the drop-down list. This opened the New Cluster Wizard..



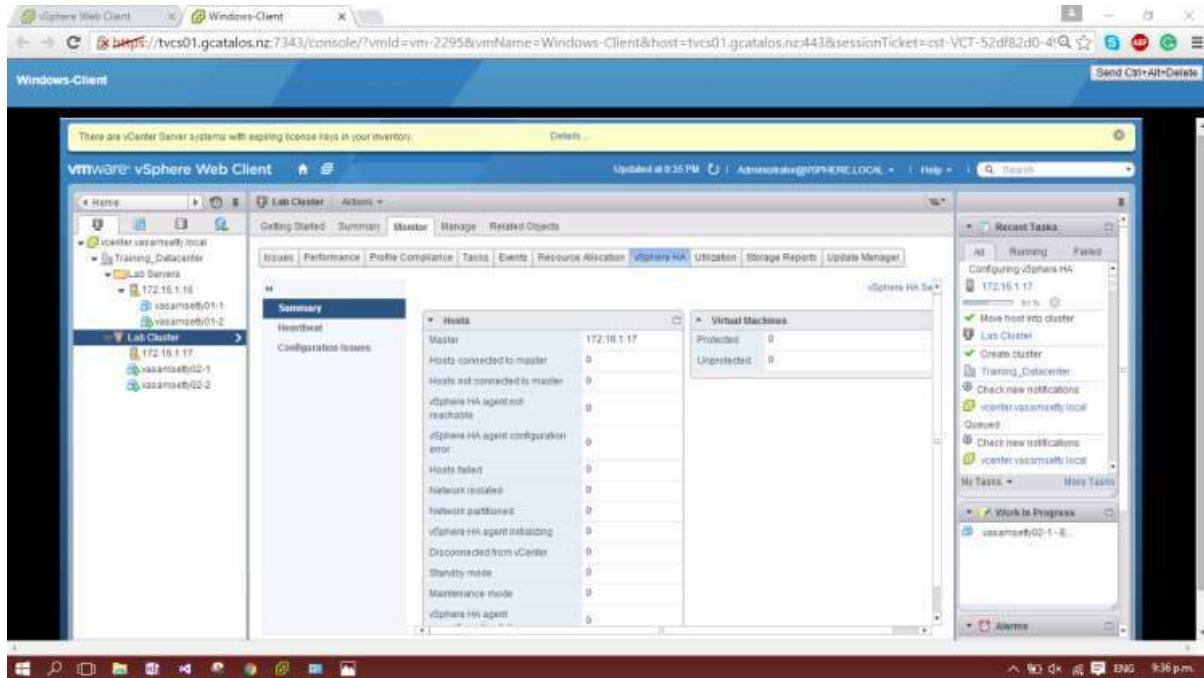
From here, I entered in 'Lab Cluster' for the cluster name, ticked the 'Turn ON' tick box for the vSphere HA, clicked the 'Percentage of cluster resources reserved as failover space capacity' radio button, and then clicked OK.

The new cluster now shows in the inventory.

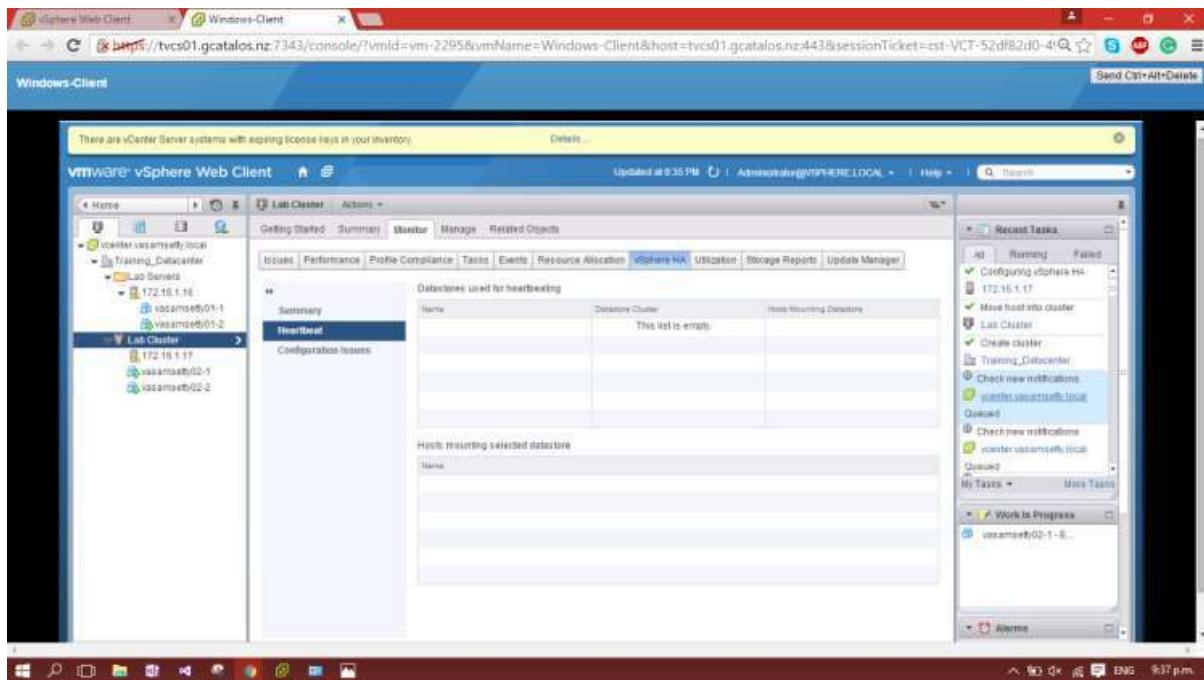


## Adding Your ESXi Host to a Cluster

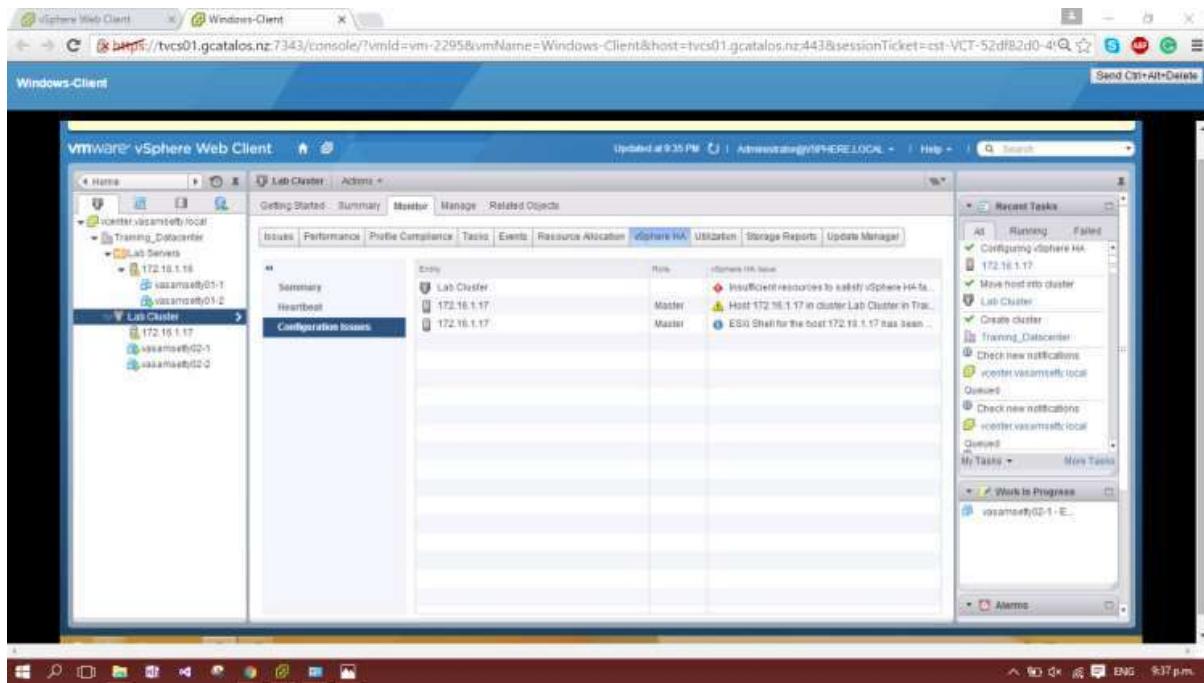
For this task I navigated to 'Home -> vCenter->Hosts and Clusters', selected one of my two ESXi host and dragged it to the new 'Lab Cluster' cluster. I then selected the new cluster and navigated to the 'Monitor -> vSphere HA' tab. My primary ESXi host (the 172.16.1.17 one) is listed as the master host, but there is no information about whether the other two virtual machines are protected or not.



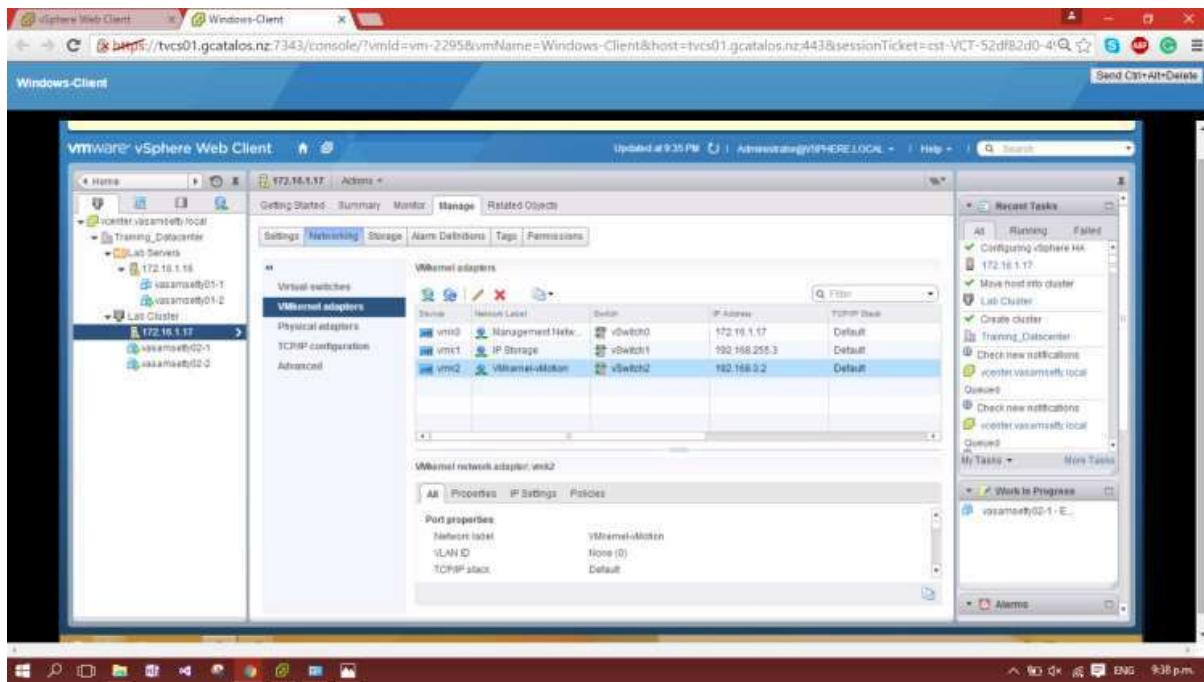
I then clicked on the 'Heartbeat' tab and found that there are currently no datastore in use for heartbeating.



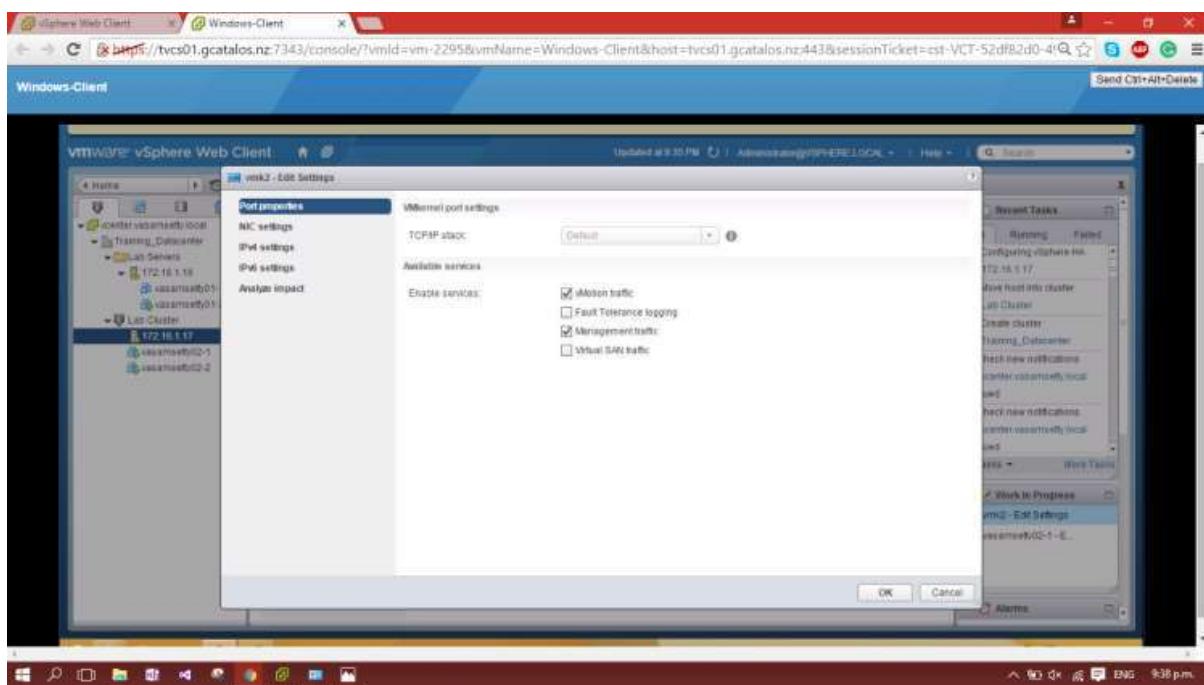
Finally, I navigated to the configuration issues tab and noticed a few errors



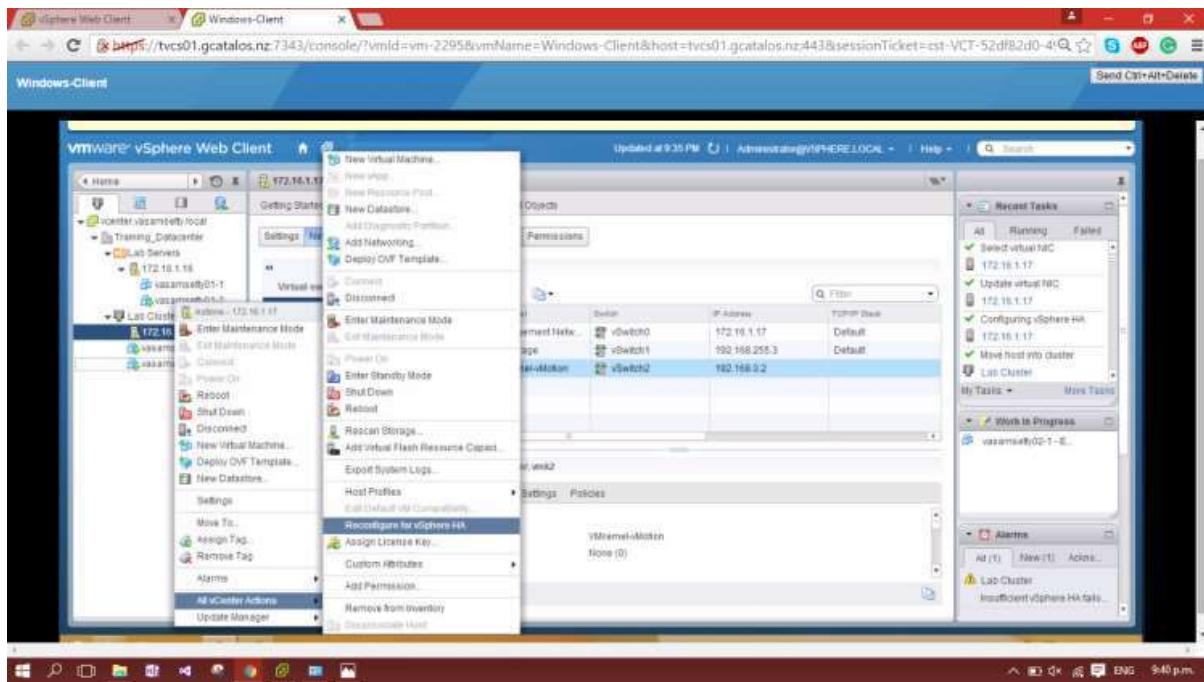
I then selected my ESXi host within the cluster, navigated to 'Manage -> Networking -> VMkernel Adapters', selected my 'VMkernel-vMotion' VMkernel adapter, and then clicked on the 'Edit Settings' button.



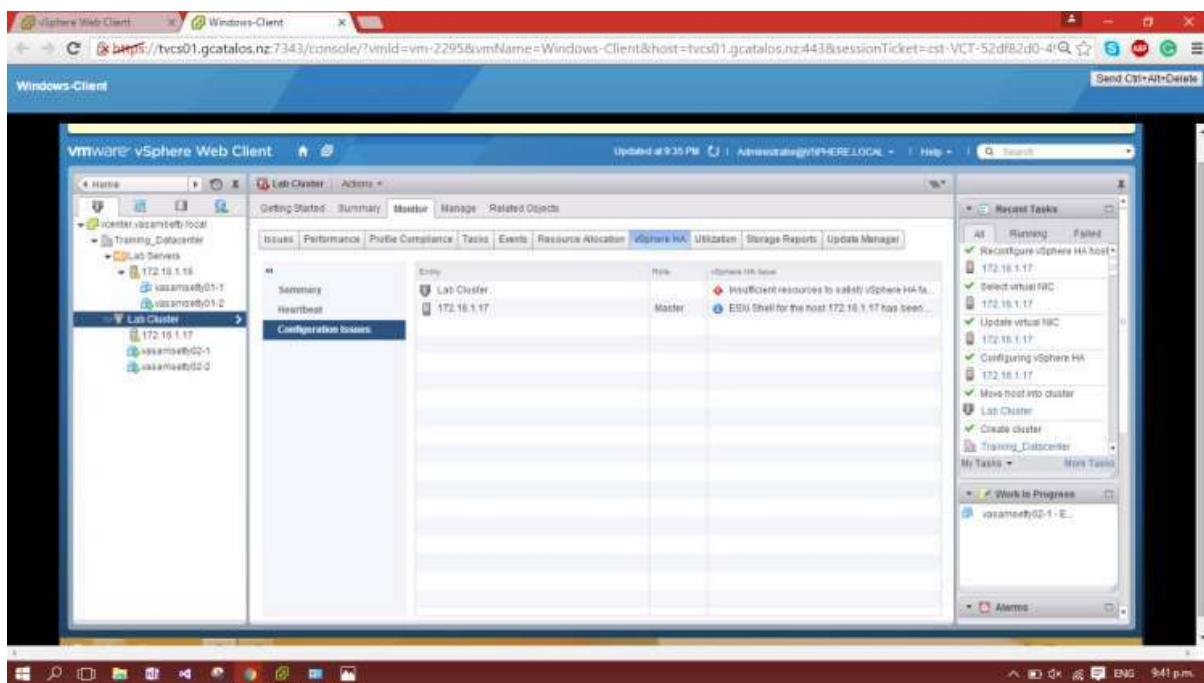
Once in the settings menu, I ticked the 'Management traffic' tick box and clicked OK



I then right-clicked my ESXi host within the cluster and selected 'All vCenter Actions -> Reconfigure for vSphere HA'



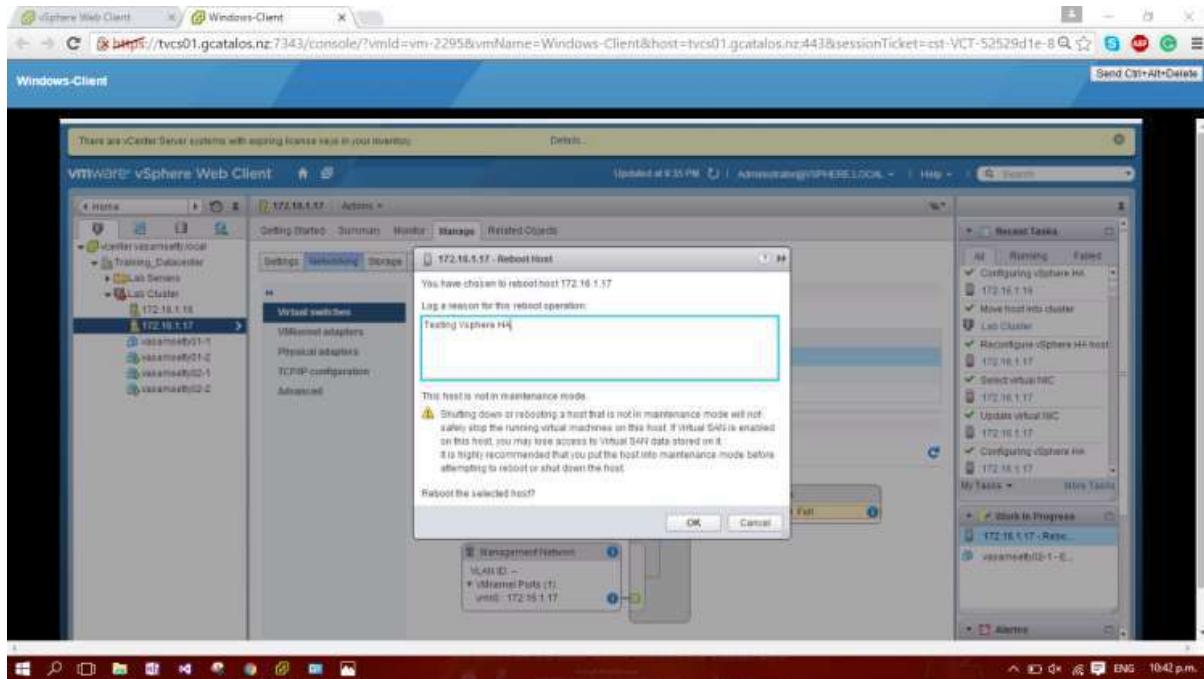
After the reconfiguration had finished, the issue regarding the management network had disappeared.



## Testing vSphere HA Functionality

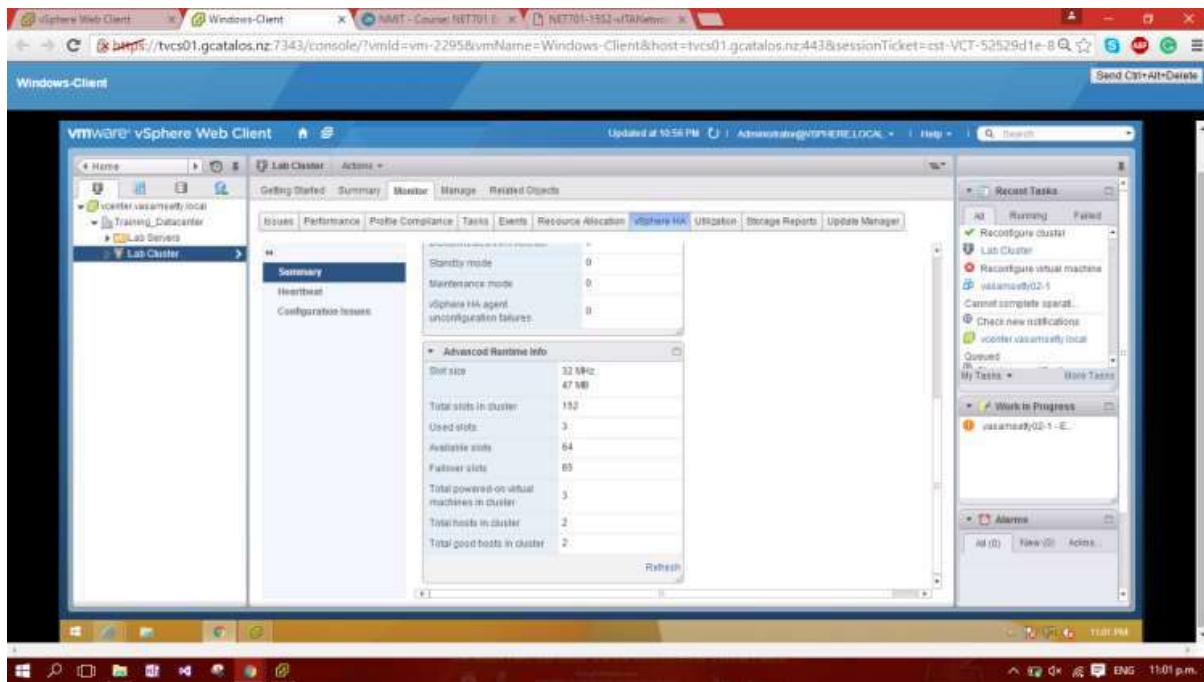
I began this task by navigating to 'Home -> vCenter -> Hosts and Clusters', selecting the master ESXi host in the cluster, navigating to the 'Related Objects -> Virtual Machines' tab, and noting the names of the virtual machines. I then simulated a host

failure by rebooting my master ESXi host – the 172.16.61.17 host. I right-clicked the host and selected ‘Reboot...’ from the drop-down menu. I was then presented with the Reboot Host menu. I entered in ‘Testing vSphere HA’ as a reason for the reboot and clicked OK.



The host initiated a reboot. I then selected the ‘Lab Cluster’ cluster and navigated to the ‘Monitor -> Events’ tab. From here, a variety of messages were displayed. Of particular note was the one regarding a ‘possible failure of a host’ which was just due to the reboot.

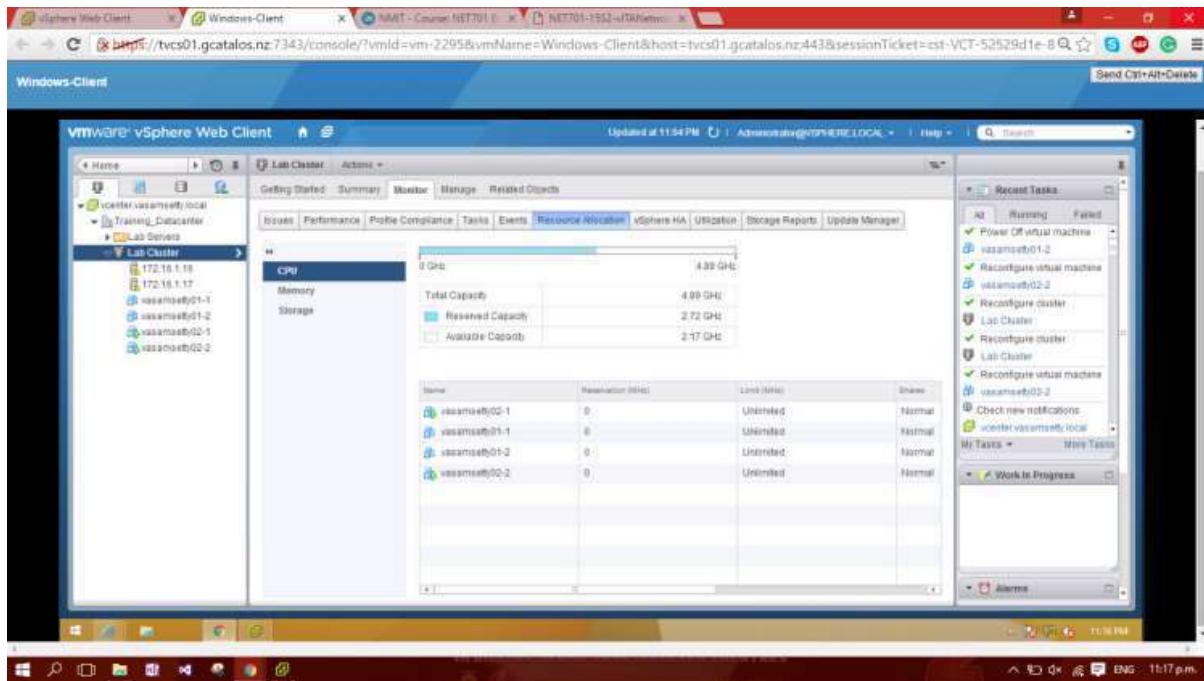
I then selected the ‘Lab Cluster’ cluster and navigated to ‘Monitor -> vSphere HA -> Summary’. I noticed that the IP address displayed for the Master had changed to the IP of my second ESXi host.



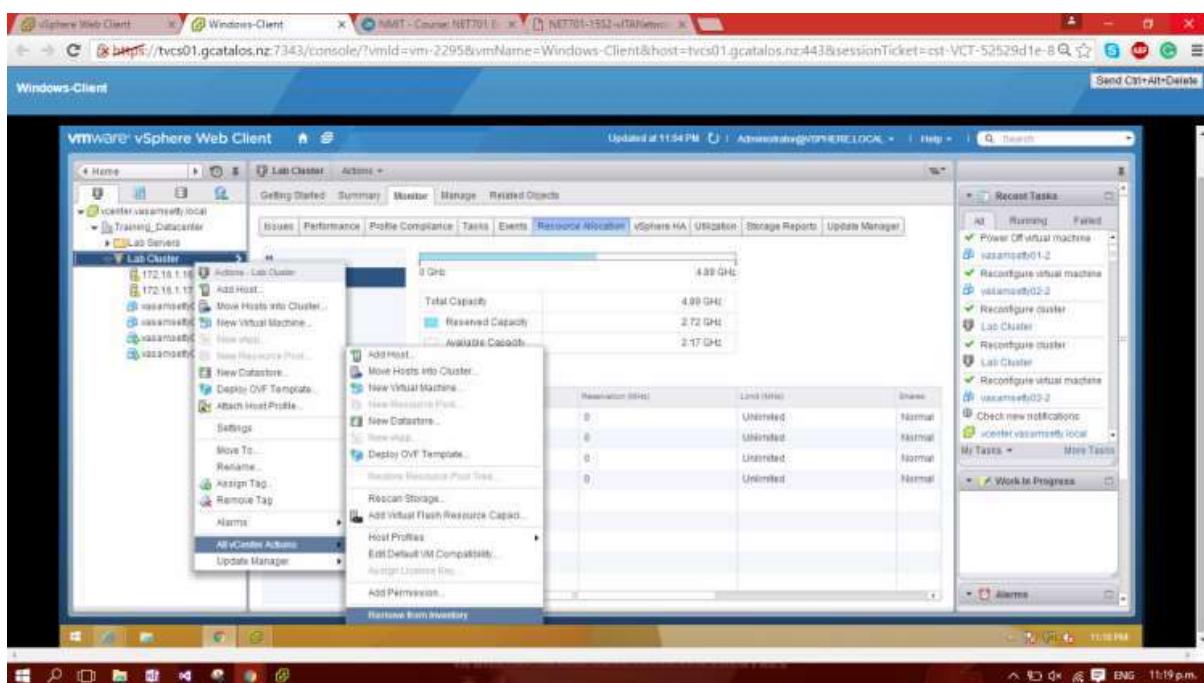
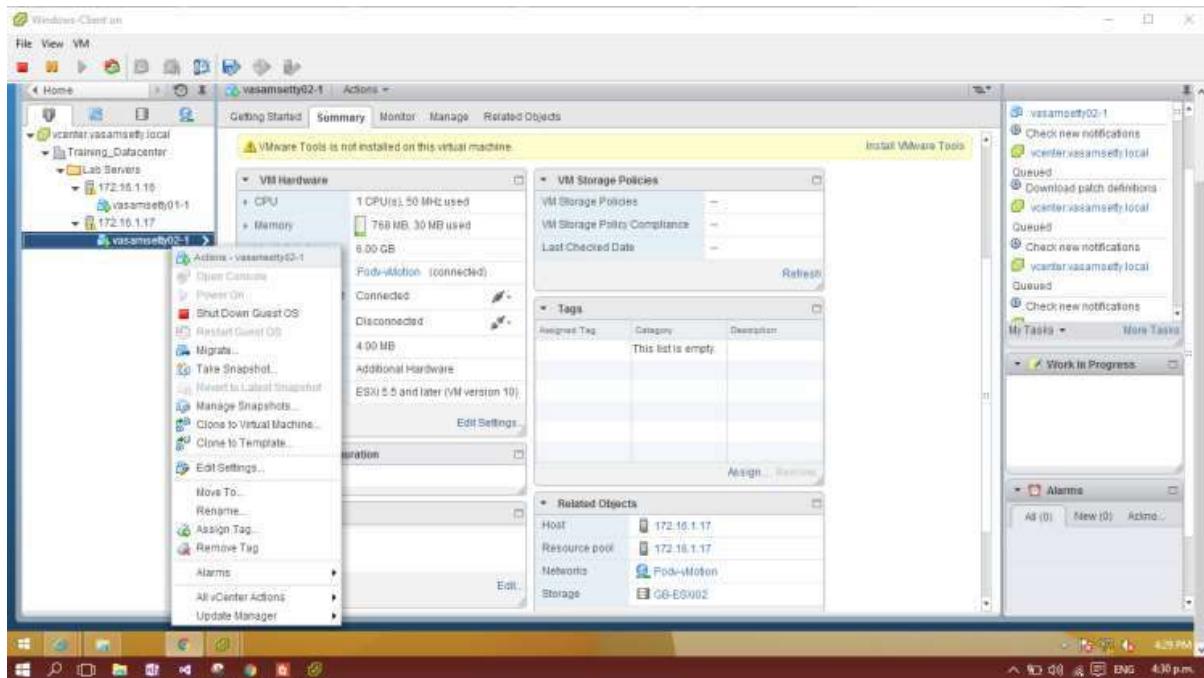
**NOTE:** The alarm saying that my first ESXi host was not responding resolved itself after 2-3 refreshes.

### Determining the vSphere HA Cluster Resource Usage

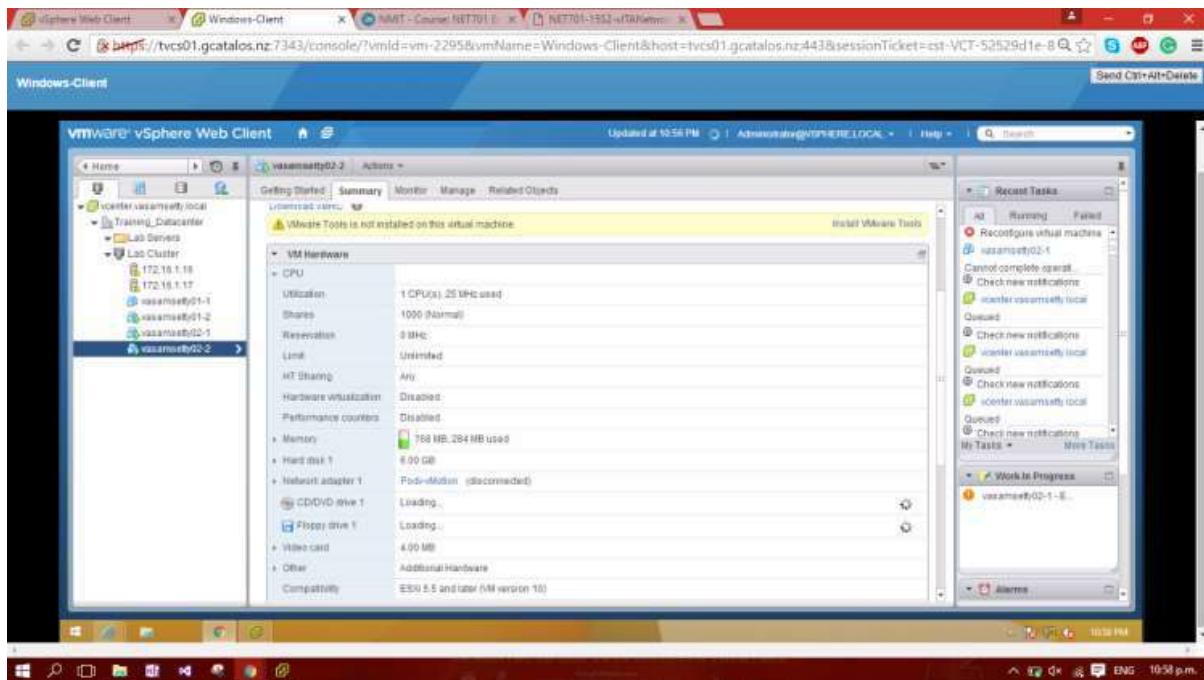
To start this task, I navigated to 'Home -> vCenter -> Hosts and Clusters', selected the 'Lab Cluster' cluster, and navigated to the 'Monitor -> Resource Allocation' tab. From here, I could see information regarding the usage of all of the resources within the cluster. Of particular interest were the CPU and Memory tabs.



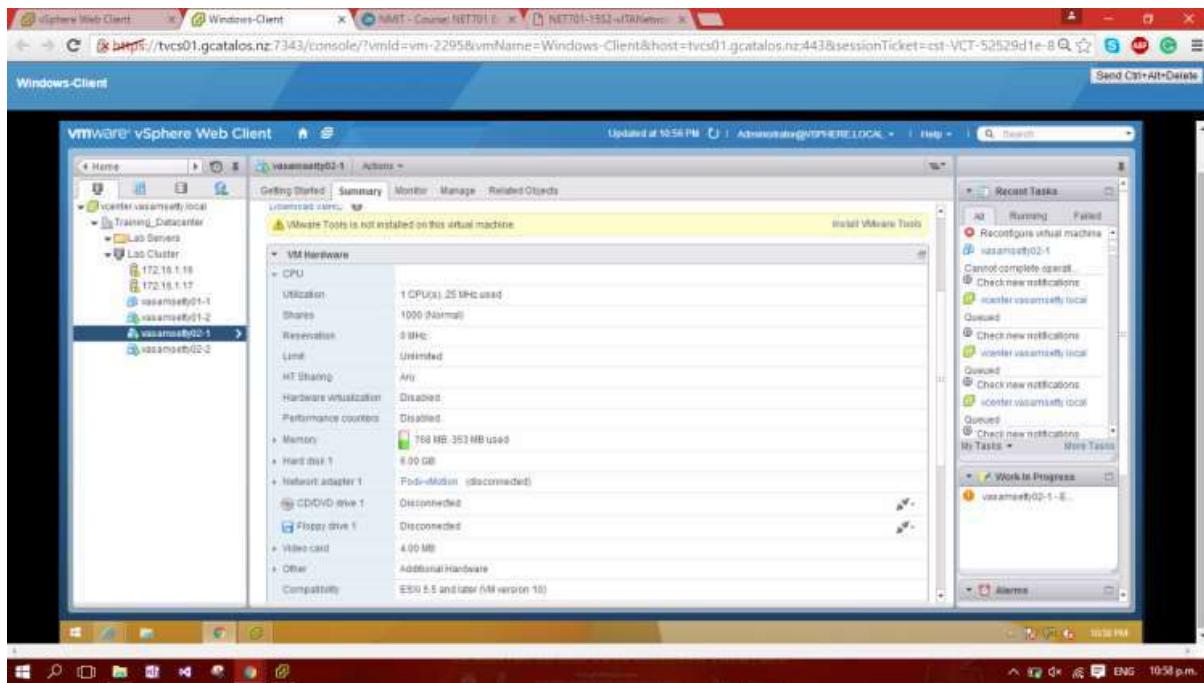
I then investigated the CPU and memory usage of my two virtual machines to determine which of the two was using the most resources. Resource usage for 'Vasamsetty02-1' virtual machine:



I then investigated the CPU and memory usage of my two virtual machines to determine which of the two was using the most resources. Resource usage for 'Vasamsetty02-1' virtual machine:



Resource usage for 'Vasamsetty02-2' virtual machine:

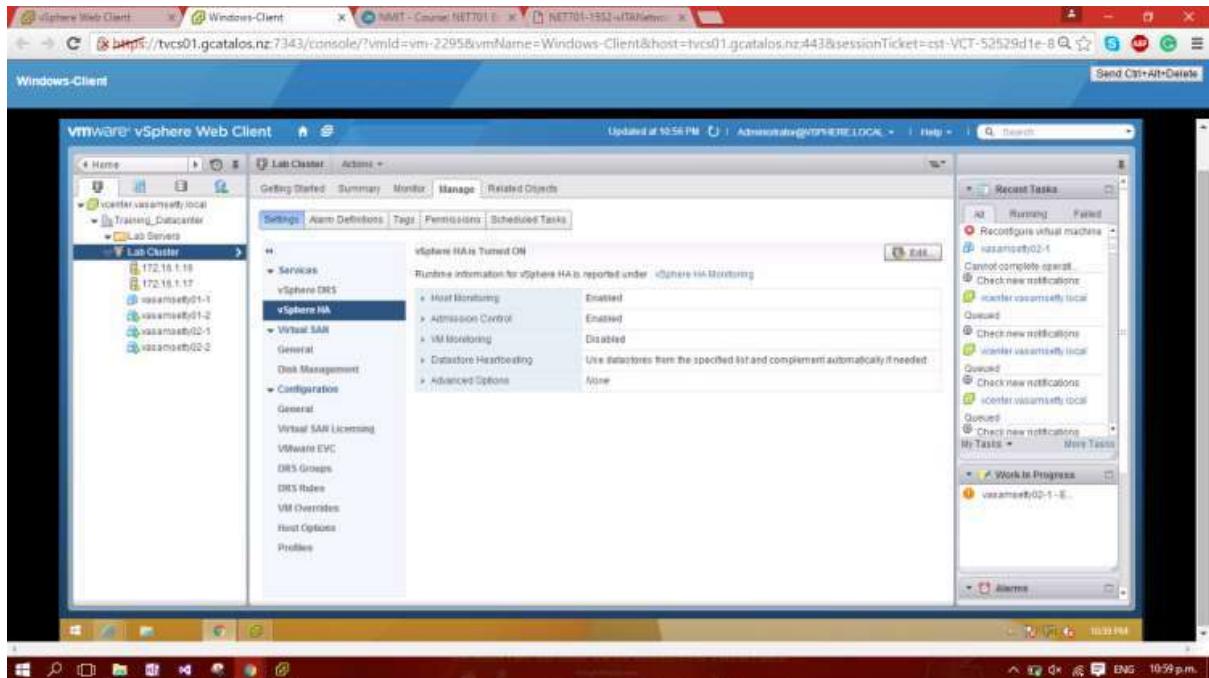


It appears that the 'Vasamsetty02-2' virtual machine is using more memory than the 'Vasamsetty02-1' virtual machine, but they are both using the same amount of CPU power.

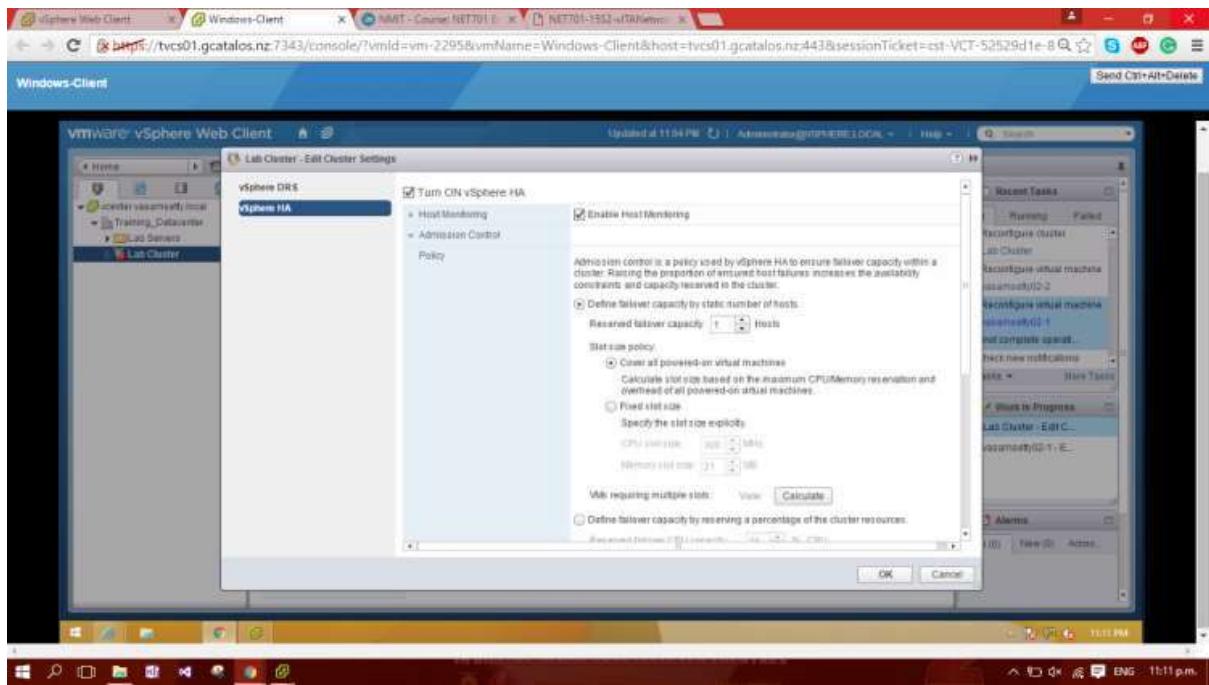
### Managing vSphere HA Slot Size

I began this task by navigating to 'Home -> vCenter -> Hosts and Clusters', right-clicking the 'Lab Cluster' cluster, and selecting 'Settings' from the drop-down menu.

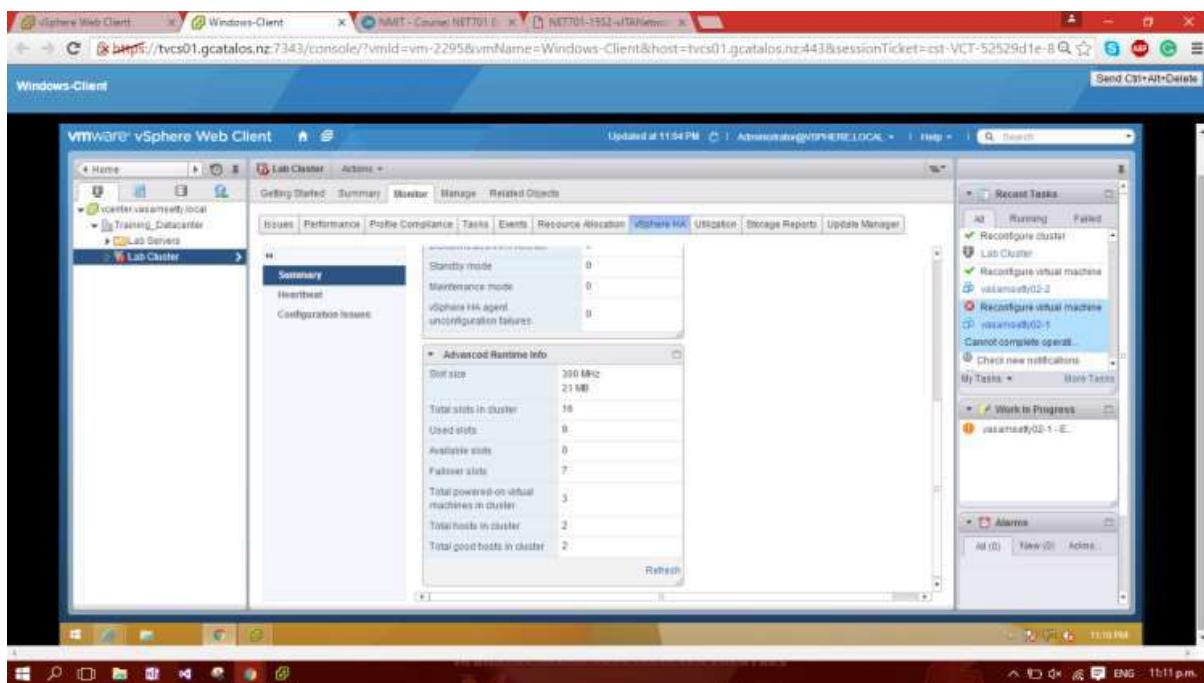
From there, I selected ‘vSphere HA’ from under ‘Services’, and clicked the ‘Edit...’ button. This opened the ‘Edit Cluster Settings’ wizard.



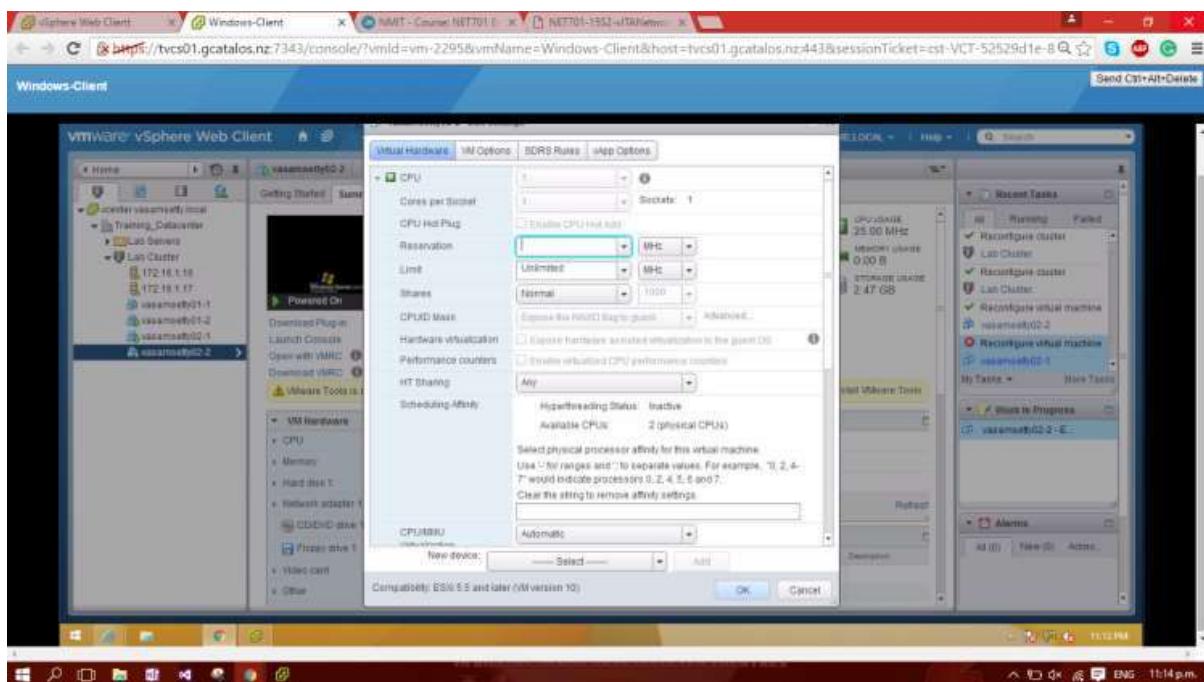
I then expanded the ‘Admission Control’ menu, changed the policy to ‘Define failover capacity by static number of hosts’, and confirmed the changes.



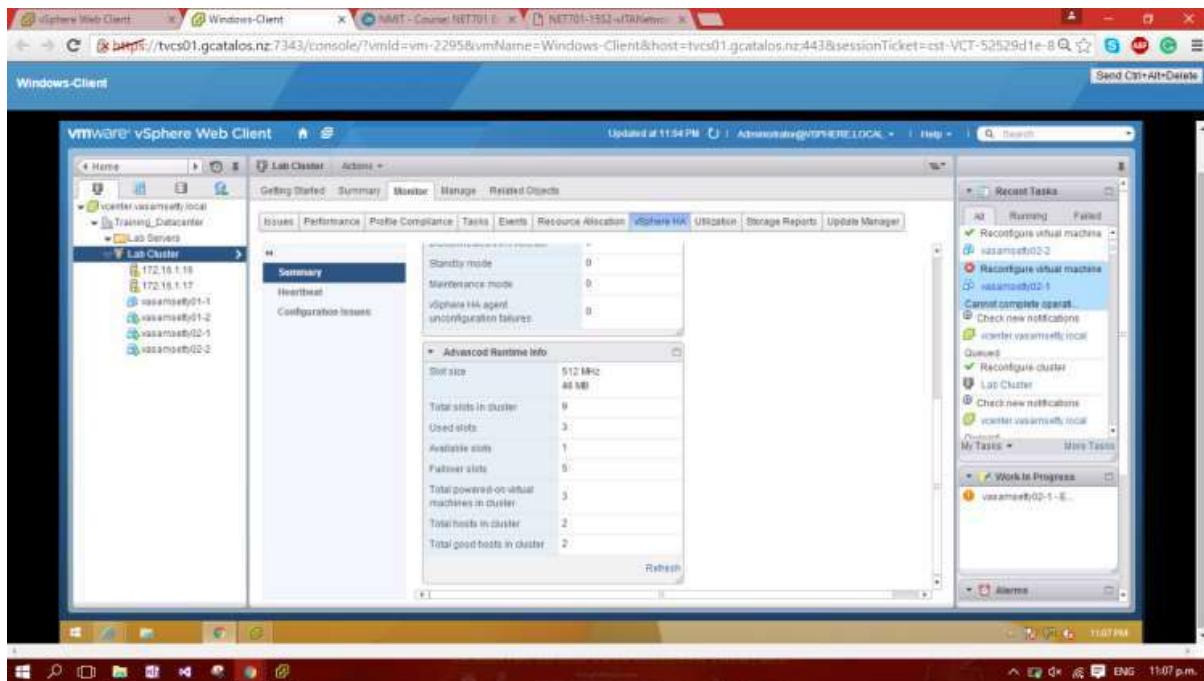
From there, I selected the ‘Lab Cluster’ cluster, navigated to the ‘Monitor -> vSphere HA -> Summary’ tab, and reviewed the information regarding resources that are required by the default slot.



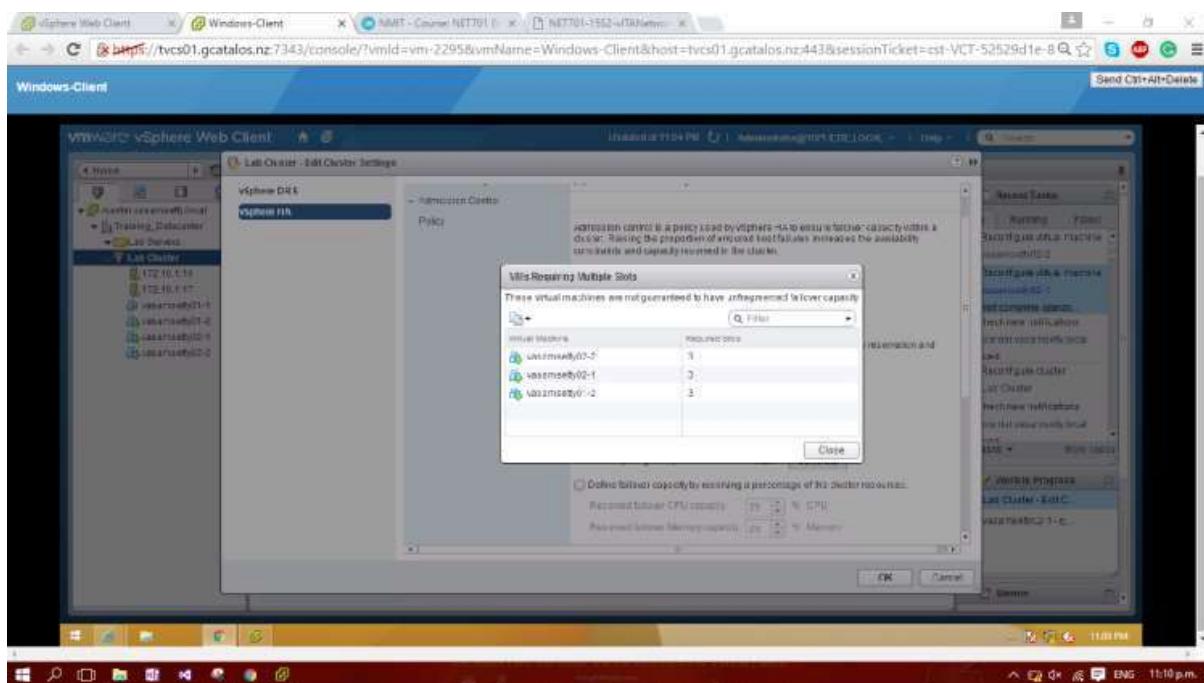
I then navigated to 'Home -> vCenter -> VMs and Templates', right-clicked my '**Vasamsetty02-2**' virtual machine, and selected 'Edit Settings...' from the drop-down menu in order to change the CPU reservation for the virtual machine. I set the CPU reservation to be 512MHz and then confirmed the change.



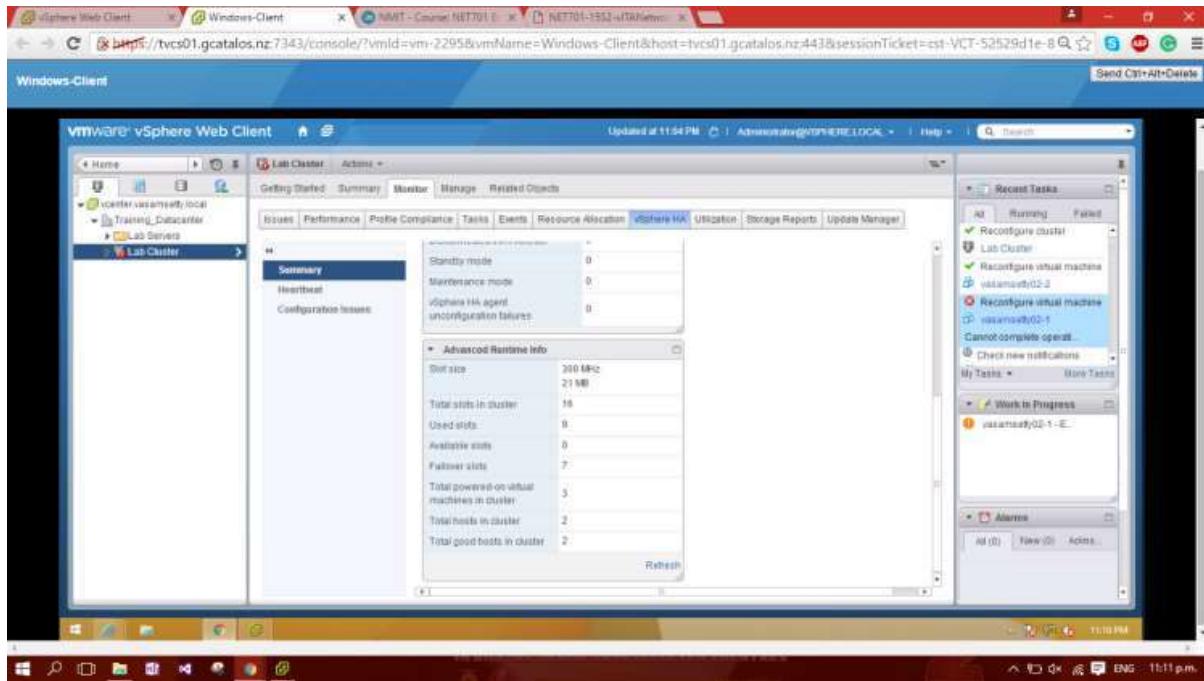
I then navigated back to 'Home -> vCenter -> Hosts and Clusters' , selected the 'Lab Cluster' cluster, navigated to 'Monitor -> vSphere HA -> Summary' and observed the change in slot CPU requirements. It now requires 512MHz rather than 32MHz.



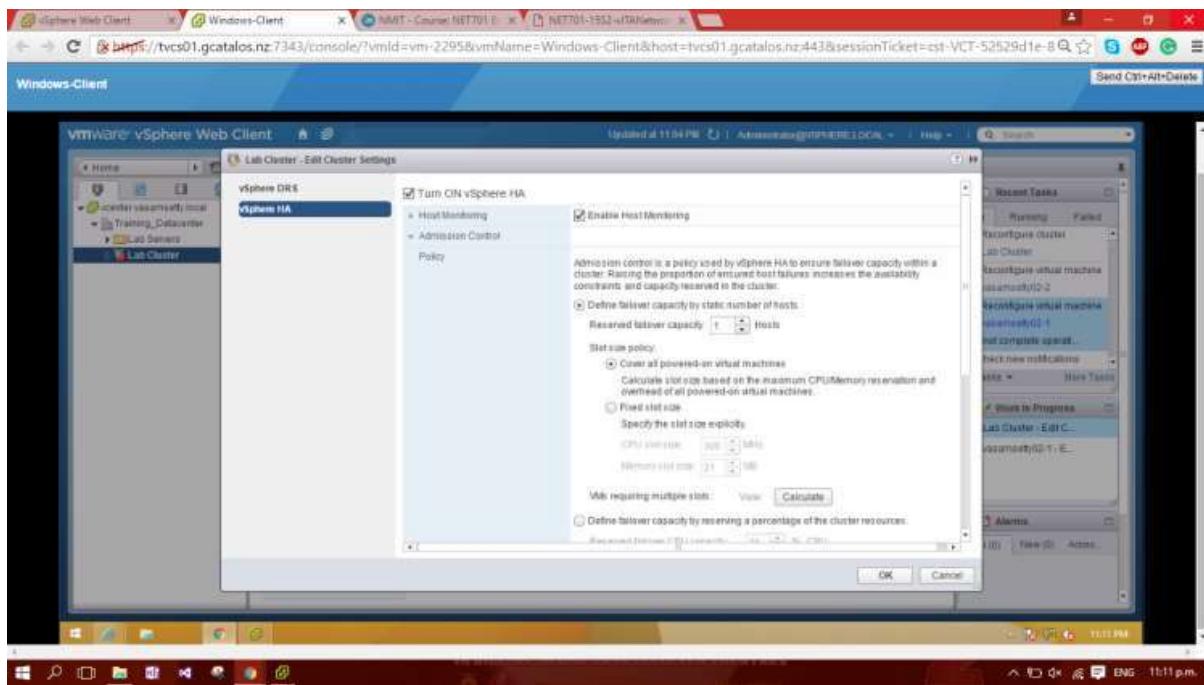
I then clicked on the 'View' link in order to see the virtual machine that required multiple slots. Sure enough, it was the '**Vasamsetty02-2**' virtual machine with the reservation requiring two slots.



I then navigated back to the 'Monitor -> vSphere HA -> Summary' tab and noticed that the CPU value for the slot size had changed to 300MHz.

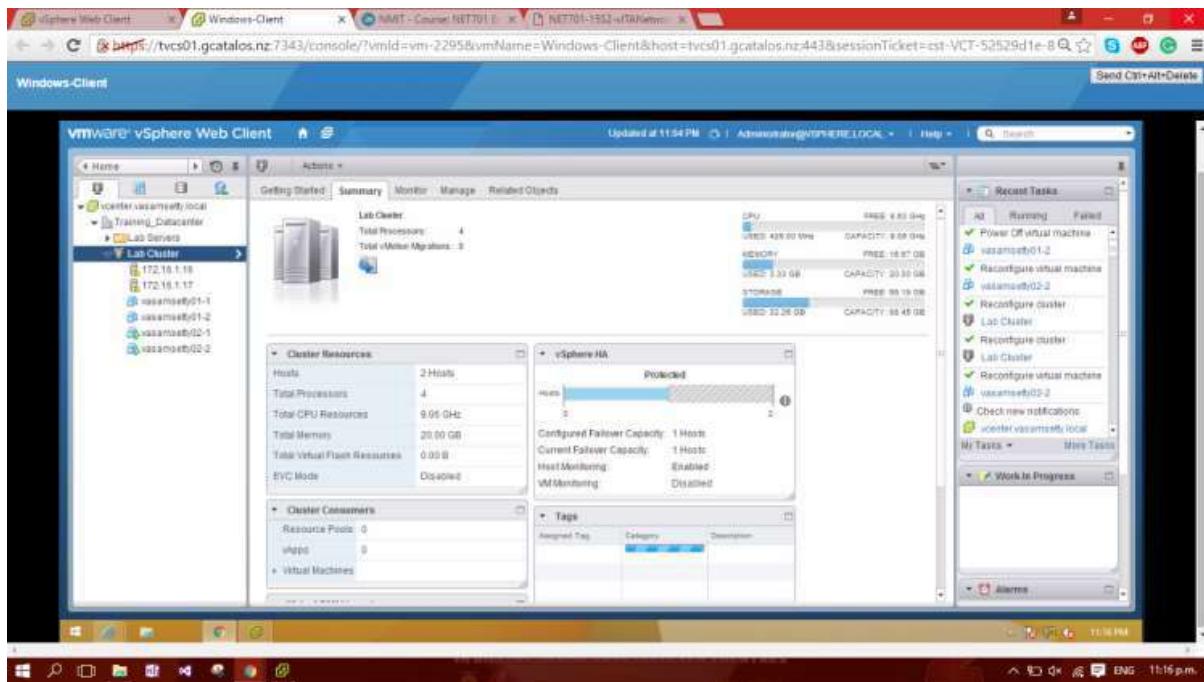


I then changed the slot size policy back to 'Cover all powered-on virtual machines...

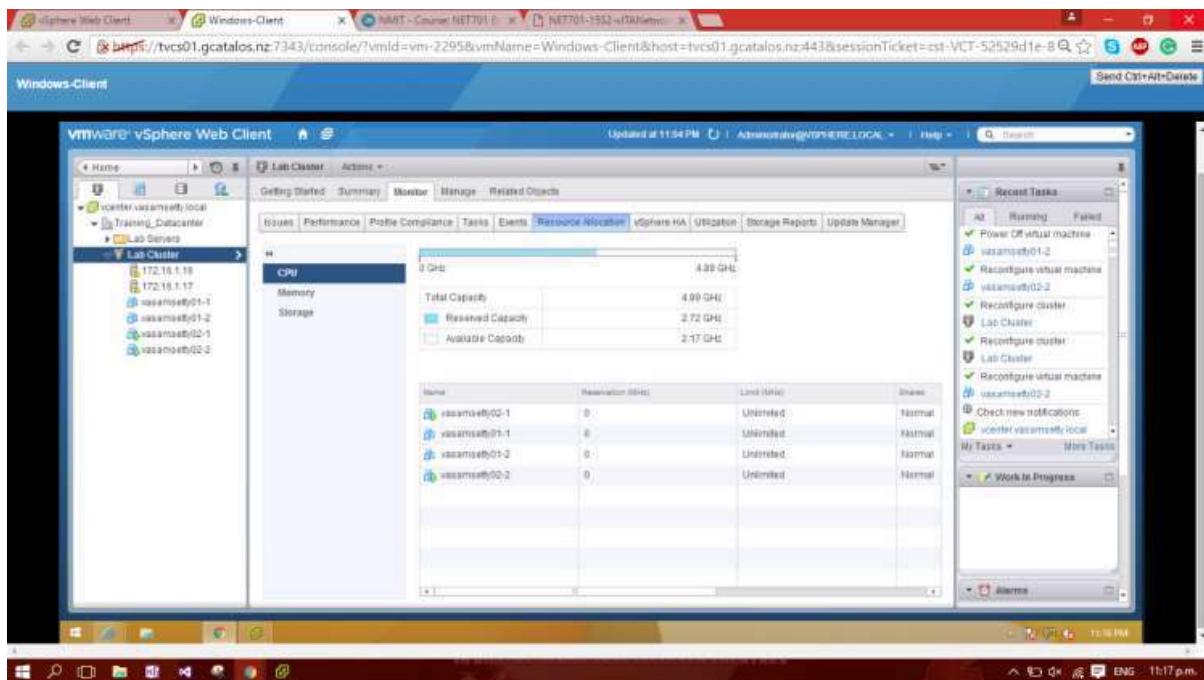


### Configuring a vSphere HA Cluster with Strict Admission Control

I started off this task by navigating to 'Home -> vCenter -> Hosts and Clusters -> Lab Cluster -> Related Objects -> Virtual Machines' and shutting down all of my virtual machines. After the two virtual machines had been powered off, I navigated to the 'Summary' tab and took note of the total memory capacity for the ESXi hosts.



There is a total of 8GB of memory available and two ESXi hosts. This therefore means that there is a total of 4GB available for each ESXi host. I noticed that only 5.51GB of the RAM is free however. This is most likely down to the ESXi hosts needing some, the cluster taking some (perhaps?), and other settings potentially using some. I then switched to the 'Monitor -> Resource Allocation -> Memory' tab and examined the capacity values. The fact that only 2.42GB of memory is available (rather than the expected 4GB) is likely due to the ESXi host itself needing some to function.



I then right-clicked my '**Vasamsetty02-1**' virtual machine, selected 'Edit Settings...', and added a 384MB memory reservation.

I then selected the 'Lab Cluster' cluster from the inventory, navigated to the 'Monitor -> vSphere HA' tab, and examined the total number of slots in the cluster. Interestingly, no slots were shown; only a value of N/A. I assume this is because the virtual machines are not currently running, therefore, no slots are needed/used.

I then powered on my '**Vasamsetty02-2**' virtual machine and refreshed the Advanced Run time Info section once again. There is now one less available slot as the other virtual machine is using one.

