Dağıtık Sistemlere Giriş

1. SSH (Secure Shell)

Güvenli bir şekilde uzaktaki bir makineye giriş yapmak ve komut çalıştırmak için kullanılan protokoldür. Amaç uzaktaki sunuculara güvenli bağlantı sağlamak, komut satırı üzerinden işlemler gerçekleştirmek.

Bash Komutu	ssh kullanıcı_adı@sunucu_adresi
Örnek Komut	ahmet@192.168.1.10

2. SCP (Secure Copy)

SSH protokolünü kullanarak dosya veya dizinleri güvenli bir şekilde kopyalamak için kullanılan komuttur. Amaç dosya transferleri yaparak yerelden uzak sunucuya veya uzak sunucudan yerel makineye dosya kopyalamak.

Bash Komutu	Scp [dosya veya seçenekler] kaynak hedef
Örnek Komut 1	scp dosya.txt ahmet@192.168.1.10:/home/alice/ (Yerel -> uzak sunucuya)
Örnek Komut 2	scp ahmet@192.168.1.10:/home/alice/dosya.txt /yerel/dizin/ (Uzak -> yerel)

3. rsync

Dosya ve dizinlerin senkronize edilmesi ve yedeklenmesi için kullanılan, verimli ve hızlı bir kopyalama aracıdır. Amaç dosya ve dizinleri yerel veya uzak makineler arasında senkronize etmek ve sadece değişiklikleri güncellemek.

Bash Komutu	rsync [seçenekler] kaynak hedef
Örnek Komut	rsync -avz /yerel/dizin/ alice@192.168.1.10:/uzak/dizin/

`-a` : Arşiv modu, dosya izinleri, zaman damgaları ve simlinkler gibi özellikleri korur.
`-v` : Ayrıntılı (verbose) çıktı verir.
`-z` : Aktarım sırasında veriyi sıkıştırır.

Uygulama 1: Aşağıda, yukarıda verilen temel Linux komutları ile örnek bir uygulama verilmiştir.

1. Adım	Bash Terminali

2.Adım	Kullanıcı ve Sunucu Bilgileri:
	REMOTE_USER="user" # Uzak sunucu kullanıcı adı
	REMOTE_HOST="192.168.1.100" # Uzak sunucu IP adresi
	REMOTE_PORT=22 # SSH portu (standart 22, farklı ise ayarlayın)

3.Adım	SSH: Uzak Sunucu Bağlantısını Test Etme: echo "1. SSH bağlantısını test ediyorum..." ssh -p $REMOTE_PORT $REMOTE_USER@$REMOTE_HOST "echo 'Uzak sunucuya başarıyla bağlandınız!'" || { echo "SSH bağlantısı kurulamadı."; exit 1; }
4. Adım	SCP: Yerel Dosyayı Uzak Sunucuya Kopyalama: LOCAL_FILE="/home/user/dosya.txt" REMOTE_DIR="/home/user/" echo "2. SCP ile dosya transferi yapılıyor..."

	scp -P $REMOTE_PORT $LOCAL_FILE $REMOTE_USER@$REMOTE_HOST:$REMOTE_DIR || { echo "Dosya transferi başarısız."; exit 1; } echo "Dosya başarıyla kopyalandı."
5. Adım	rsync: Dizin Senkronizasyonu: (Yerel ve uzak dizin tanımlamaları) LOCAL_DIR="/home/user/dizin/" REMOTE_SYNC_DIR="/home/user/dizin/" echo "3. rsync ile dizin senkronizasyonu başlatılıyor..." rsync -avz -e "ssh -p $REMOTE_PORT" $LOCAL_DIR $REMOTE_USER@$REMOTE_HOST:$REMOTE_SYNC_DIR || { echo "rsync işlemi başarısız."; exit 1; } echo "Dizin senkronizasyonu tamamlandı."
6.Adım	Sonuç: echo "Tüm işlemler başarıyla tamamlandı."

Alternatif : Script ile çalıştırma

1. Script Oluşturma: Yukarıdaki kodu bir metin düzenleyiciye yapıştırın ve `example.sh` adıyla kaydedin.

2. Çalıştırılabilir hale getirin: Terminali açın ve aşağıdaki komutu girin

Komut	chmod +x example.sh

3. Scripti Çalıştırın : Terminalde aşağıdaki komut ile scripti başlatın

Komut	./example.sh

Uygulama 2: Python ile basit bir **istemci-sunucu modeli uygulaması**

* Uygulama basit bir istemci (client) ve sunucu(server)'dan oluşmakta sunucu belirli bir port üzerinden bağlantıları dinlerken, istemci bu sunucuya bağlanır, mesaj gönderir ve sunucudan gelen yanıtı alır.

server.py

```
import socket

def start_server():
    host = '127.0.0.1'  # Yerel makine adresi
    port = 65432        # Dinlenecek port numarası

    # TCP soketi oluşturma
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Soket, belirtilen host ve port'a bağlama
    server_socket.bind((host, port))
    # Gelen bağlantıları dinleme (maksimum 1 bağlantı için)
    server_socket.listen(1)
    print(f"Sunucu {host}:{port} üzerinde dinlemede...")

    # Gelen bağlantı kabul etme
    conn, addr = server_socket.accept()
    print(f"Bağlantı kabul edildi: {addr}")

    # Bağlantı üzerinden veri alışverişi
    with conn:
        while True:
            data = conn.recv(1024)  # 1024 byte'a kadar veri al
            if not data:
                # Veri alınamadığında (bağlantı kesildiğinde) döngüden çıkma
```

	break
	print("Gelen veri:", data.decode())
	# Gelen veriyi geri gönder (echo)
	conn.sendall(data)
	
	server_socket.close()
	print("Sunucu kapatıldı.")
	
	if __name__ == '__main__':
	start_server()

client.py	import socket
	
	def start_client():
	host = '127.0.0.1' # Bağlanılacak sunucu adresi
	port = 65432 # Sunucunun dinlediği port
	
	# TCP soketi oluşturulma
	client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
	# Sunucuya bağlantı kuruluyor
	client_socket.connect((host, port))
	
	# Gönderilecek mesaj
	message = "Merhaba, TCP!"
	client_socket.sendall(message.encode())
	print("Gönderilen mesaj:", message)
	
	# Sunucudan gelen yanıtı alma
	data = client_socket.recv(1024)
	print("Sunucudan gelen:", data.decode())
	
	client_socket.close()
	
	if __name__ == '__main__':
	start_client()

Uygulamayı Çalıştırma:

1. Terminalden aşağıdaki komutu çalıştırın

Terminal kodu	python server.py

2. İkinci bir terminal açıp aşağıdaki kodu çalıştırın

Terminal kodu	python client.py