```
const { Token, TOKEN_PROGRAM_ID } = require("@solana/spl-token");
const { clusterApiUrl, Connection, Keypair } = require('@solana/web3.js');

// Replace with your actual NOWNodes API key
const NOWNODES_API_KEY = 'your_nownodes_api_key_here';

// Set up the connection to Solana RPC via NOWNodes
const connection = new Connection(
    `https://sol.nownodes.io/${NOWNODES_API_KEY}`,
    'confirmed'
);

// Create a new token
async function createToken() {
    const payer = Keypair.generate(); // This keypair will act as the payer for the transaction
    const mintAuthority = Keypair.generate(); // Authority that can mint new tokens
    const freezeAuthority = Keypair.generate(); // Authority that can freeze token accounts

    const token = await Token.createMint(
        connection,
        payer,
        mintAuthority.publicKey,
        freezeAuthority.publicKey,
        9, // Decimal places, matching the Solana CLI's default
        TOKEN_PROGRAM_ID
    );

    console.log("Token Mint Address:", token.toBase58());
    return { token, payer, mintAuthority, freezeAuthority };
}

// Mint some tokens
async function mintTokens(token, destinationAccountPublicKey, amount, payer, mintAuthority) {
    // Specify the mint public key, the destination account, and the mint authority
    const mintToInstruction = Token.createMintToInstruction(
        TOKEN_PROGRAM_ID,
        token,
        destinationAccountPublicKey,
        mintAuthority.publicKey,
        [],
        amount
    );

    const transaction = new solanaWeb3.Transaction().add(mintToInstruction);
    await connection.sendTransaction(transaction, [payer, mintAuthority]);

    console.log(`Minted ${amount} tokens to ${destinationAccountPublicKey.toBase58()}`);
}

// Check the balance of a token account
async function checkBalance(publicKey) {
    let balance = await connection.getTokenAccountBalance(publicKey);
    console.log(`Balance: ${balance.value.uiAmount}`);
}

// Swap tokens
async function swapTokens(token1, token2, amount, payer, mintAuthority) {
    // Get the token accounts for the two tokens
    const token1Account = await Token.getOrCreateAssociatedAccountInfo(
        connection,
        payer,
        token1,
        mintAuthority.publicKey
    );

    const token2Account = await Token.getOrCreateAssociatedAccountInfo(
        connection,
        payer,
        token2,
        mintAuthority.publicKey
    );

    // Swap the tokens
    const swapInstruction = Token.createTransferInstruction(
        TOKEN_PROGRAM_ID,
        token1Account.address,
        token2Account.address,
        payer.publicKey,
        [],
        amount
    );

    const transaction = new solanaWeb3.Transaction().add(swapInstruction);
    await connection.sendTransaction(transaction, [payer]);

    console.log(`Swapped ${amount} tokens from ${token1.toBase58()} to ${token2.toBase58()}`);
}

async function main() {
    const { token, payer, mintAuthority, freezeAuthority } = await createToken();

    const destinationAccount = Keypair.generate();
    await mintTokens(token, destinationAccount.publicKey, 100, payer, mintAuthority);

    await checkBalance(destinationAccount.publicKey);

    const token2 = await createToken().then(({ token }) => token);
    await swapTokens(token, token2, 50, payer, mintAuthority);
}

main();
```